Etude et mise en oeuvre d'une architecture pour l'authentification et la gestion de documents numériques certifiés : application dans le contexte des services en ligne pour le grand public

Mahamat Ahmat Abakar

▶ **To cite this version:**

Mahamat Ahmat Abakar. Etude et mise en oeuvre d'une architecture pour l'authentification et la gestion de documents numériques certifiés : application dans le contexte des services en ligne pour le grand public. Autre [cs.OH]. Université Jean Monnet - Saint-Etienne, 2012. Français. NNT : 2012STET4019 . tel-00975965

HAL Id: tel-00975965

https://theses.hal.science/tel-00975965

Submitted on 19 May 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Thèse

Pour obtenir le grade de

Docteur de l'université Jean Monnet de Saint-Étienne

Génie Informatique

Présentée et soutenue publiquement le 22 novembre 2012

Par

ABAKAR Mahamat Ahmat

Étude et mise en œuvre d'une architecture pour l'authentification et la gestion de documents numériques certifiés

Application dans le contexte des services en ligne pour le grand public.

Directeur de thèse :
Jacques FAYOLLE
Co-Directeur de thèse :
Mikaël ATES

Composition du jury :

Lionel BRUNIE	Professeur,	INSA de Lyon	*Rapporteur*
Laurent VERCOUTER	Professeur,	INSA de Rouen	*Rapporteur*
Mikaël ATES	Docteur,	Entr'ouvert	*Examinateur*
Jacques FAYOLLE	Professeur,	TELECOM SAINT-ETIENNE	*Directeur*
Jean Jacques ROUSSEAU	Professeur,	Université de Saint-Étienne	*Président*

Remerciements

Le Gouvernement de la République du Tchad a adopté ces dernières années une stratégie d'éducation et de formation. L'objectif de cette stratégie vise à apporter une qualité de formation et accroitre de façon significative le nombre des enseignants dans tous les niveaux de formation. L'Institut Universitaire des Sciences et Techniques d'Abéché (IUSTA) est l'une des meilleures institutions du supérieur du Tchad qui œuvre dans ce sens. Actuellement un nombre significatif est en pleine formation de master ou de thèse à travers plusieurs universités partenaires. C'est ainsi que j'ai été accueilli d'abord en master web intelligence puis en thèse à l'Université Jean Monnet de Sain-Etienne. La formation étant financée par une bourse octroyée par la coopération Tchad-France.

Je tiens à remercier Jean Jacques Rousseau qui est parmi le pilier de cette coopération entre la France et Tchad. Il a beaucoup œuvré et a contribué à la formation des plusieurs cadres du supérieur du Tchad qui occupent actuellement des très hautes responsabilités dans tous les secteurs. Il m'a d'abord accueilli en stage de perfectionnement, puis en stage de master recherche et enfin durant tous mes travaux de thèse au sein de son laboratoire LT2C. Jean Jacques est non seulement un acteur de la coopération, mais un conseiller, un guide qui ne cesse de nous suivre, nous orienter, et prodiguer des conseils, reproches, remarques et suggestions. Reçois encore mes sincères remerciements pour l'honneur que tu m'as fait en acceptant de présider le jury. Je remercie chaleureusement mon ex-Directeur Mahamout Youssouf Khayal pour son dynamisme, sa motivation et ses efforts pour la formation des enseignants de l'IUSTA. Mes remerciements vont également à mon Directeur Idriss Yacoub Halawlaw et le Sécrétaire Général Daboulaye Djimoudjebaye pour leur encouragement durant mon cursus au sein de l'IUSTA.

Je remercie mon Directeur de Thèse Jacques Fayolle pour son accueil au sein de l'équipe SATIN, pour ses conseils, suggestions et orientation tout au long de mes travaux en Master et en thèse.

Je remercie sincèrement Mikaël Ates pour le suivi, ses conseils et encouragements. Mikaël m'a suivi depuis ma formation en Master jusqu'à l'aboutissement de cette thèse. Tous ses conseils et son orientation ont été constructifs pour moi. Il est non seulement un encadrant mais un collègue avec qui nous avons partagé des moments inoubliables soit autour d'un café ou dans un terrain de sport. Les mots me manquent pour exprimer ma profonde gratitude et reconnaissance.

Je remercie Laurent Vercouter et Lionel Brunie pour avoir accepté d'être les rapporteurs de mon mémoire. Leurs remarques et suggestions ont été constructives et ont permis de clarifier certains points.

Je remercie mes collègues de l'équipe SATIN Christophe, Jeremy et Gilbert avec qui j'ai passé des moments de courtoisies et échanger des idées constructives. Je remercie également Yves Gael, Benjamin, Antoine, Pierre Olivier, Jule et Merriem. Je ne pouvais oublier Frederique Laforeste pour ses remarques.

Je remercie tous les collègues du labo LT2C sans exception Jean Pièrre, Bernard, Eric, Chantal, Marie Françoise, Béatrice, Damien, François, Bruno, Stéphane, Eric, Ali, Didier, Faouzi, Hadi, Kekesi, Elie, Badreddine, Imail, Zana, Bilal, Fariz, Souad,…, la liste est longue

Je remercie tous mes compatriotes avec qui j'ai passé des moments de retrouvailles : Amir, Kriga, Désiré, Frank, Soudy, Farikou, Yaya, Mahamat, Ahmat, Khamis, Koularambaye, Lamaï, Arafat. Zaki, Kitio, Jérôme et sa famille, ...

Je remercie toute ma famille, ma femme Achta, mes enfants Mahamat Ahmat et Limane pour leur patience durant cinq années de perturbation. J'espère le moment est venu pour rattraper les temps de solitude.

Je remercie Dieu de m'avoir aidé et donné le courage et la patience pour aboutir à ce résultat.

Résumé

Dans un environnement ouvert tel que l'Internet, les interlocuteurs sont parfois inconnus et toujours dématérialisés. Les concepts et les technologies de la confiance numérique et de la sécurité informatique doivent se combiner pour permettre un contrôle d'accès en environnement ouvert.

Dans nos travaux, nous nous proposons d'étudier les concepts majeurs de cette problématique, puis de concevoir, et enfin de développer un système fonctionnel, basé sur des standards du contrôle d'accès, pour un environnement ouvert et appliqué à l'Internet. Plus précisément, notre étude consiste à mettre en œuvre une architecture de contrôle d'accès basée sur la confiance numérique.

L'élément central de cette architecture est l'environnement utilisateur très riche et déployé en ligne. Cet environnement est doté de trois modules principaux qui permettent à l'utilisateur de mener à bien ses transactions. Ces modules sont le module d'analyse de règlements, le module de récupération de données et le module de validation de règlements. Nous avons élaborés des algorithmes utilisés dans ces modules.

L'usage est le suivant. L'utilisateur demande un service à un fournisseur de services, celui-ci analyse la requête de l'utilisateur et extrait le règlement à partir de la base des règles de contrôle d'accès. Cette architecture est conçue à l'aide de modèles de contrôle d'accès basé sur les attributs et le langage XACML. Ce règlement contient des conditions à satisfaire par l'utilisateur pour obtenir le droit d'accès à la ressource demandée. Le module d'analyse de règlement permet à l'utilisateur d'analyser le règlement reçu du fournisseur de service. Cette analyse consiste à vérifier à l'aide d'un algorithme la disponibilité de ses informations auprès de ses sources d'information d'identité de confiance pour le fournisseur de services. Le module de récupération de données permet ensuite à l'utilisateur de récupérer ses certificats. Le module de validation lui permet de tester qu'il satisfait le règlement grâce aux certificats. Si le règlement est satisfait l'utilisateur diffuse ses certificats au fournisseur de service.

La conception de ce système repose sur un ensemble de brique technologiques étudiées et décrites dans ces travaux. Ce document débute par une étude des différents cas d'usage dans le domaine des transactions en ligne. Cette étude permet de mettre en évidence la problématique de la gestion des identités numériques en environnement ouvert. Les organisations virtuelles, la notion de partenariat et la confiance sont des éléments clés qui entrent dans la conception des systèmes de contrôle d'accès basé sur la confiance. Une première étude d'un ensemble de modèles de contrôle d'accès nous permet de dégager le modèle ABAC et le langage XACML pour la conception de notre système. Dans un second temps, nous concevons le modèle de données de notre système de contrôle d'accès distribué et nous présentons les algorithmes. Ensuite, nous concevons une architecture protocolaire satisfaisant les besoins d'interopérabilité entre les différentes entités impliquées. Il s'agit de protocoles permettant d'établir une session auprès d'un système, permettant de véhiculer un règlement de contrôle d'accès et permettant d'obtenir et de diffuser des informations entre tiers de confiance. La dernière partie est consacrée à l'implémentation réalisée en langage python et en utilisant le « framework » de développement Web Django.

Table des matières

INTRODUCTION

INTRODUCTION

Motivation

Les sociétés modernes sont aujourd'hui fortement dépendantes des technologies de l'information et des communications. En outre, le concept de réseaux en environnement ouvert, tel que l'Internet, permet par certains aspects la constitution de nouvelles sociétés régies par des règles souvent bien éloignées de celles qui régissent ces mêmes sociétés.

La notion de « Cloud computing » rassemble sous un même nom un ensemble de concepts dans lesquels on retrouve la notion d'environnement ouvert. Établir une relation, contractuelle ou de confiance, est ainsi différente sur Internet que dans la vie civile. Pour autant, ces relations sont tout aussi importantes, que ce soit pour exister dans un réseau social que pour conduire des transactions administratives ou de e-commerce.

Ainsi, le contrôle d'accès en environnement ouvert pose la problématique de donner un accès à un algorithme que l'on souhaite piloté intentionnellement. Pour cela, on va par exemple s'assurer que l'acteur puisse être tracé en cas de litige, ou l'on va chercher à obtenir une contrepartie en échange du service. Les mécanismes d'authentification peuvent servir ce type de besoins. Cependant, le sujet demandeur du service n'est généralement par connu du système d'information hébergeant le fournisseur de service. Il est donc dans ce cas nécessaire de s'appuyer sur des tiers de confiance qui connaissent le sujet et vont émettre des assertions le concernant.

En résumé, dans un environnement ouvert, les interlocuteurs sont parfois inconnus et toujours dématérialisées. Les concepts et technologies de la confiance numérique et de la sécurité informatique vont se compléter pour permettre de réaliser un contrôle d'accès en environnement ouvert. Dans ces travaux, nous nous proposons d'étudier les concepts majeurs de cette problématique, puis de concevoir et enfin de développer un système fonctionnel, basé sur divers standards de contrôle d'accès, pour un environnement ouvert et appliqué à l'Internet.

Dans notre document le terme « environnement ouvert » se définit comme suit [Ates, 2009] : « Un environnement où il ne peut exister d'autorité centrale régissant toutes les mises en relation entre entités qui le peuplent, ni tous les accès que ces dernières requièrent. C'est un environnement dans lequel diverses entités, potentiellement inconnues, ont à établir des relations qui visent à satisfaire les conditions de contrôle d'accès des fournisseurs de services et à donner à l'usager des moyens de contrôle et de confiance envers les organisations. Le Web au travers de l'Internet est une implémentation qui constitue un tel environnement. » Dans un environnement

ouvert, l'utilisateur exploitant les technologies Web a besoin de mobilité et peut être amené à changer régulièrement de terminal. On suppose donc que son terminal est simplement muni d'un navigateur Web et qu'aucune donnée n'y est stockée.

Contexte de la thèse & Problématique

Un utilisateur inconnu du fournisseur de service obtient un accès au travers de la confiance du fournisseur de service envers des organisations attestant d'informations sur l'utilisateur. Avec l'évolution de la technologie de l'information et de la communication, et principalement du « cloud computing », nous nous sommes posés la question de comment la confiance numérique entre organisations sera utilisée dans l'avenir pour permettre à des usagers d'accéder à des services en ligne offerts par des organisations desquelles ils sont inconnus ? Avec les atouts offerts par la numérisation des informations, l'opportunité est d'exploiter ces nouvelles technologies pour le bien être de la société. C'est ainsi que nous avons opté pour l'exploitation des systèmes de gestion des identités numériques dans le cadre de la fourniture de services sensibles (transaction administratives, achats, etc.) à des utilisateurs inconnus par des organisations diverses. L'objet de cette thèse est « l'étude et la mise en œuvre d'une architecture pour l'authentification et la gestion de documents numériques certifiés appliquée dans le contexte des services en ligne pour le grand public ». Nous cherchons à travers ces travaux à étudier et à réaliser un système de gestion d'identité pour le contrôle d'accès dans un environnement ouvert. Il s'agit de concevoir un système de contrôle d'accès permettant de décrire les conditions d'accès des utilisateurs préalablement aux demandes d'accès, conditions d'accès basées sur la confiance [Ates & Abakar, 2011].

Nous nous focalisons sur l'usager qui, dans le cadre de ses activités « en ligne » dans la sphère privée, souhaite consommer un service Web soumis à un contrôle d'accès et s'appuie sur diverses organisations pour obtenir des certificats pour ensuite les présenter au fournisseur de services.

Nous fixons deux contraintes majeures à ce système qui correspondent aux besoins actuels :

- Le sujet n'appartient pas à une unique organisation mère comme c'est le cas dans de nombreux travaux. On suppose ici qu'il est connu de multiples organisations et peut donc obtenir des certificats de ces multiples organisations afin d'obtenir des droits auprès d'organisations tierces. En effet, il est courant de voir des travaux sur l'interopérabilité où il est fait la supposition qu'il suffit à l'utilisateur d'appartenir à une unique organisation partenaire et de présenter un unique certificat, contenant potentiellement des attributs, pour obtenir l'accès.

- Lors de la conception de ce système, nous nous appuierons autant que possible sur des modèles de contrôle d'accès existants dans la mesure où ils permettent d'exprimer les liens de confiance entre organisations. Nous nous appuierons également sur des normes ou spécifications standards ouvertes existantes pour bâtir l'architecture protocolaire.

Cas d'usage

Afin de faire converger nos travaux vers un système fonctionnel, nous décrivons dans la suite un cas d'usage qui servira d'illustration tout au long de cette thèse.

Nous avons choisi ce cas qui met en jeu la confiance numérique, l'exploitation d'attributs d'identité et le contexte du sujet.

Le cas d'usage est « La phase d'enregistrement et de paiement en ligne pour une location de voiture ». Dans ce cas d'usage, il s'agit d'autoriser l'accès lorsque l'on a une preuve de l'identité du sujet permettant d'engager des poursuites judiciaires en cas de litige. Il s'agit également de s'assurer de plusieurs conditions d'accès au service : la possession d'un permis de conduire et d'une assurance, tout deux valides. Enfin, il s'agit bien sûr de réaliser le paiement.

Nous supposerons pour ce cas d'usage qu'il existe un tiers de confiance représentant l'État qui va émettre en ligne des certificats servant en quelque sorte de pièces d'identité dématérialisée. Ces certificats contiennent un identifiant aussi appelé pseudonyme. Ainsi, le loueur de voiture pourra en cas de litige obtenir l'identité réelle du sujet par révocation du pseudonyme, identité réelle connue de l'État. Un autre tiers de confiance nécessaire, également supposé hébergé par une administration, est celui qui est en charge d'émettre des certificats représentant des permis de conduire valides. Ces certificats ne sont émis que si le conducteur a toujours des points. Le sujet doit être couvert par une assurance. Celle-ci doit également émettre en ligne des certificats correspondant à des attestations d'assurance dématérialisées.

Nous faisons également la supposition d'une architecture de confiance à grande échelle permettant à la société loueuse de voiture de pouvoir reconnaître les signatures numériques de ces organisations tierces.

Organisation du mémoire

Cette thèse est structurée en quatre chapitres :

Le premier chapitre est consacré à l'analyse du contexte et de l'état de l'art. L'objet de cette analyse est de déterminer le choix des modèles à utiliser dans nos travaux. Nous commençons par une première étude sur les organisations virtuelles. Il est présenté les concepts des organisations

virtuelles, la notion de partenariat entre organisations, la notion de confiance et le principe de la fédération. Ensuite sont abordés les systèmes de contrôle d'accès. Nous présentons les modèles de contrôle d'accès classiques traditionnels (DAC, MAC, IBAC, RBAC, ABAC, OrBAC, ...) et les modèles de contrôles d'accès distribués et collaboratifs. Nous clôturons ce premier chapitre par une analyse sur le lien entre le contrôle d'accès et la confiance. Nous dégageons de ce chapitre la pertinence du modèle de contrôle d'accès basé sur les attributs (ABAC – Attribute-Based Access Control) pour nos travaux. En effet, notre système s'appuie que sur des tiers de confiance habilités à certifier et gérer les données d'un sujet, données décrivant principalement les attributs de l'identité d'un sujet.

Le deuxième chapitre est consacré à l'étude de notre système de contrôle d'accès. Ce chapitre traite principalement de la description fonctionnelle d'un système centré sur l'usager afin qu'il soit possible de gérer ses informations d'identités certifiées au sein de transactions numériques. Il s'agit de mettre en évidence l'ensemble des composants (l'environnement utilisateur, le fournisseur de services, les sources de données) ainsi que leur fonctionnement. Nous débutons par la description des phases de traitement d'une requête d'accès à un service. Ensuite, nous mettons en évidence les moyens utilisés lors d'un contrôle d'accès à un service afin de catégoriser les différents types d'informations exploitées et de caractériser les conditions du contrôle d'accès en environnement ouvert. Plusieurs algorithmes peuvent s'appliquer au traitement des règlements de contrôle d'accès basés sur les attributs. Pour pouvoir ensuite présenter ces algorithmes, nous présentons le modèle de données décrivant notre modèle de contrôle d'accès élaboré. Ce modèle est présenté sous la forme d'un diagramme de classes. Nous présentons ensuite les différents algorithmes utilisés pour les traitements des règlements de contrôle d'accès. Le premier algorithme (algorithme 1) utilisé au niveau de module d'analyse de données (coté utilisateur) permet à l'utilisateur de vérifier la disponibilité auprès de ses sources des données nécessaires à la satisfaction des conditions du règlement. Les deux autres algorithmes permettent respectivement la vérification des prédicats d'une règle logique portant sur des attributs d'identité (algorithme 2) et l'évaluation des expressions logiques (algorithme 3).

Le troisième chapitre traite de l'architecture protocolaire de notre système. Nous présentons d'une part les relations entre les différentes organisations et d'autre part leurs éléments (modules) constitutifs. Il s'agit de montrer comment les organisations à travers leurs entités (modules) collaborent, échangent des informations pour atteindre un objectif commun. Nous répondons notamment aux questions suivantes. Quels sont les protocoles utilisés pour établir une session auprès d'un système ? Comment véhiculer un règlement de contrôle d'accès ? Comment diffuser les

informations d'identité ? Nous traitons pour cela trois points dans ce chapitre. Le premier point porte sur les bases de l'architecture de confiance, notamment sur les concepts de l'authentification unique (Web SSO), sur la prise en charge des attributs certifiés et non certifiés, et sur les architectures de délégation de l'autorisation standard. Le second point est relatif à l'utilisation des espaces de nom et la standardisation des documents, notamment concernant les documents de règlement de contrôle d'accès et les documents de description des sources de données. Enfin, nous définissons les différents schémas architecturaux, notamment les protocoles permettant la déclaration de sources d'attributs auprès de l'environnement utilisateur, l'envoi du règlement par le fournisseur de service et l'envoi des certificats par l'utilisateur.

Le quatrième et dernier chapitre est consacré à l'implémentation de notre système. L'implémentation est faite en python et à l'aide du « framework » de développement d'application Web Django. Nous implémentons le modèle de données et les algorithmes présentés au chapitre deux. Nous décrivons ensuite un algorithme permettant de convertir les règlements reçus au format XACML en objets de notre modèle. Enfin, nous décrivons les points principaux de l'implémentation des composants Web de l'architecture, notamment les plans d'adressages applicatifs pour les protocoles définis au chapitre trois.

CHAPITRE 1

ANALYSE DU CONTEXTE

ET ETAT DE L'ART

1 ANALYSE DU CONTEXTE ET ETAT DE L'ART

1.1 Introduction

Ce premier chapitre est consacré à l'étude des systèmes de contrôle d'accès. Nous commençons par aborder la notion de l'organisation virtuelle qui représente de nos jours une forme de gestion mettant en relation plusieurs organisations qui doivent collaborer et échanger pour atteindre un objectif commun. La notion de partenariat entre organisations est intimement liée à l'organisation virtuelle. Elle se traduit par l'établissement d'un réseau de confiance entre partenaires.

Nous traitons également dans ce chapitre de la problématique des identités numériques et de leur gestion par la fédération d'identité. La fédération d'identité [Pfitzmann, 2003] est aujourd'hui standardisée au travers d'architecture Web permettant l'échange d'informations certifiées.

Nos travaux portent sur un système de contrôle d'accès basé sur la confiance, cela implique que nous faisions également un tour d'horizon des principaux modèles de confiance et des principaux modèles de contrôle d'accès.

De ce chapitre nous déterminons les plus adaptés aux besoins de nos travaux.

1.2 Organisations Virtuelles, partenariats, fédération et gestion d'identité

L'utilisation des technologies de l'information et de la communication a conduit à la constitution d'une nouvelle forme d'organisations appelées Organisations Virtuelles (OV). Une organisation virtuelle se compose d'organisations dont les systèmes d'information sont interconnectés dans le but de répondre à des besoins de collaboration et d'échange d'information.

Plusieurs publications ont tenté de définir l'organisation virtuelle. Nous avons retenu deux définitions également relevées par Kamel Michel dans ses travaux « Patrons organisationnels et techniques pour la sécurisation des organisations virtuelles » [Kamel, 2008]. La première définition est donnée par [Camarinha-Matos & Lima, 1998] qui considèrent l'organisation virtuelle comme une alliance temporaire d'organisations (organisations, institutions, industries, etc.) qui se réunissent pour partager leurs compétences et leurs ressources afin de mieux répondre aux opportunités métiers qui se présentent. La coopération est soutenue par des réseaux informatiques. La deuxième définition est donnée par [Upton & Mcafee, 1996] qui définissent l'organisation virtuelle comme une communauté de douzaines voire de centaines d'organisations, chacune se base

sur ce qu'elle sait faire le mieux, toutes reliées par un réseau électronique qui leur permet d'opérer de façon flexible et non onéreuse, sans se soucier de leurs emplacements respectifs.

Plusieurs travaux qui traitent généralement de l'organisation virtuelle tels que ceux de [Coma, 2009], [Kamel, 2009] ou encore [Baina, 2007] font la supposition que l'utilisateur appartient à une unique organisation et qu'il lui suffit d'arriver dans une organisation partenaire en présentant un unique certificat, contenant potentiellement des attributs. Cela permet une fédération simple entre partenaire où l'utilisateur n'a qu'à choisir son organisation d'appartenance pour permettre les échanges d'information d'identité le concernant.

Nos travaux se positionnent dans un environnement plus complexe où l'utilisateur a de multiples sources. Nous supposons une organisation virtuelle faite de fournisseurs de services et de sources d'informations portant sur les identités numériques des utilisateurs. Les utilisateurs vont pouvoir obtenir des accès à divers services par la présentation d'informations issues de sources desquelles ils sont connus, sources qui sont de confiance pour les fournisseurs de services.

1.2.1 Notion de partenariat entre organisations et confiance

Le dictionnaire Larousse décrit le partenariat dans une approche systémique comme étant un système associant des partenaires. Selon [Merrini, 2001] dans « Le partenariat : Histoire et essai de définition », le partenariat est défini comme étant le minimum d'action commune négociée visant à la résolution d'un programme reconnu commun.

Nous retenons que la notion de partenariat est conjointement liée à celle de l'association. Un partenariat est le fait d'être partenaire ou associé. Il définit ainsi l'association entre deux ou plusieurs organisations qui doivent coopérer en vue de réaliser un objectif commun.

La relation liant les différents partenaires peut être formalisée par un contrat ou un protocole de collaboration [Huu & al, 2005] dans lequel le rôle et la contribution de chacune des parties sont clairement définis. Dans le cadre de notre travail, le partenariat concerne un ensemble d'organisations devant interagir, échanger, dans le but de fournir un accès sécurisé d'une entité sollicitant un service auprès d'un fournisseur de service quelconque.

Le contrat ou le protocole de collaboration régissant les parties partenaires doit permettre d'assurer la sécurité des échanges et l'interopérabilité [Yahalom & al, 1993] tout en fixant la portée de politiques de sécurité de chacune des organisations partenaires.

Ce contrat matérialise la confiance entre les partenaires. Les partenaires ainsi liés forment un

réseau de confiance.

1.2.2 La fédération et gestion d'identité

La fédération d'identité est aujourd'hui un terme qui désigne les échanges d'informations certifiées portant sur les identités numériques entre organisations.

La notion de fédération d'identité est née des difficultés rencontrées par le monde universitaire pour la gestion et le partage de leurs ressources. En effet, les ressources web d'un établissement sont destinées à son personnel, enseignants ou étudiants. Cependant, il doit être possible de partager certaines ressources avec ses partenaires avec lesquels un lien de confiance sera établi. Il s'agit en pratique d'autoriser une personne n'appartenant pas à l'établissement d'accéder aux ressources.

Plusieurs initiatives [Group Oxford, 2007] ont vu le jour parmi lesquelles Shibboleth et Liberty Alliance. Le principe de la fédération y est très similaire ce qui a permis leur convergence vers la norme SAML2 de l'OASIS. Le principe est le suivant : Lorsqu'un utilisateur demande un service auprès d'une organisation fournisseur de service, il est redirigé vers son organisation fournisseur d'identités pour s'authentifier.

En délégant ainsi l'authentification on résout les problèmes de sécurité lié à la circulation des mots de passe et on évite la multiplication des comptes d'un même utilisateur dans plusieurs organisations.

SAML est conçu pour permettre les échanges certifiés entre les différents systèmes de gestion des identités des membres d'un cercle de confiance. Il permet l'échange de certificats entre organisations à l'aide de technologie Web en assurant sécurité et respect de la vie privée.

1.2.3 Liens de confiance

Les systèmes de contrôle d'accès simples et traditionnels sont utilisés dans un environnement où les utilisateurs sont déjà connus à l'avance ainsi que leurs droits d'accès. La notion de confiance est presque inexistante dans la mesure où chaque utilisateur prouve son identité par les informations qui lui sont attribuées préalablement.

Dans un environnement ouvert, le problème lié à l'authentification et à l'autorisation se pose. Dans un tel environnement, pour donner accès à un utilisateur, plusieurs paramètres sont à prendre en compte parmi lesquels la notion de confiance. Les utilisateurs demandant l'accès ont des données

relatives à leur identité reparties à travers différents domaines que représentent les organisations partenaires. Le contrat ou protocole régissant le partenariat des différentes organisations, se base sur un modèle de confiance pour accorder l'accès à un utilisateur.

Plusieurs modèles de contrôle d'accès basé sur la confiance ont été proposés. La plupart de ces modèles établissent la correspondance directe entre la politique de contrôle d'accès et les paramètres du modèle de confiance [Hristo, 2004].

La conception d' un modèle de confiance tient compte de la source à partir de laquelle l'information est obtenue. [Lin, 2006] met en évidence un certain nombre de concepts relatifs à un modèle de confiance. Il s'agit de définir les paramètres pouvant influencer la nature des négociations et qui peuvent être directs ou indirects entre les organisations devant s'échanger des informations. La négociation peut être directe de pair à pair (figure 1.1), ou indirecte de pair vers une entité intermédiaire (figure 1.2), ou encore en passant par une fédération (figure 1.3). Ces cas sont illustrés dans les figures ci-dessous.

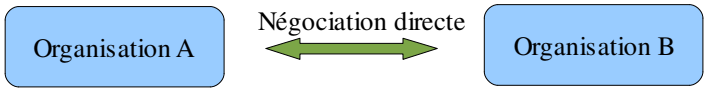

Figure 1.1: Négociation directe entre les deux organisation

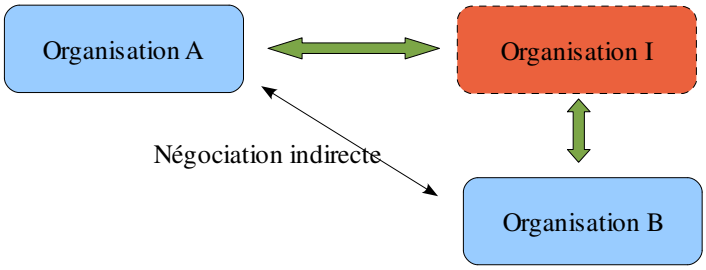

Figure 1.2 : Négociation indirecte entre deux organisations avec intermédiaire

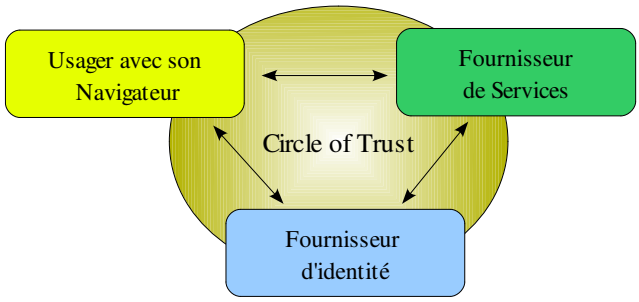

Figure 1.3 : Négociation par fédération

Sur la figure ci-dessus, l'usager requiert un service auprès d'un fournisseur de services. Le fournisseur de services ne connaît pas l'usager mais va s'appuyer sur un fournisseur d'identité de celui-ci pour obtenir des informations le concernant. C'est l'exemple type d'une fédération où une autorité de certification délivre un certificat à un usager. Nous détaillons ce concept de confiance dans la suite de nos travaux.

1.3 Les modèles de contrôle d'accès

Nos travaux impliquent un système de contrôle d'accès permettant de baser l'accès sur des certificats issus de tiers de confiance. Nous faisons ici l'état de l'art des modèles de contrôle d'accès.

Les modèles de sécurité peuvent être classés selon deux catégories principales :

- les modèles de sécurité classiques traditionnels, conçus pour gérer la sécurité d'une organisation ;
- et les modèles de sécurité collaboratifs, conçus pour gérer la sécurité d'un ensemble d'organisations qui doivent coopérer.

1.3.1 Les modèles traditionnels de sécurité

D'une manière générale, un modèle peut être défini comme un formalisme qui offre une vue subjective mais pertinente de la réalité. La modélisation améliore l'intelligibilité du système modélisé, elle permet de mettre en évidence la structure, obtenir une description du comportement, et pouvoir ainsi faire apparaître des propriétés. Les modèles de sécurité sont en ce sens un formalisme de description des politiques de sécurité.

Les politiques de sécurité formalisent les relations entre les sujets, les objets, les actions et les droits d'accès.

Les modèles de contrôle d'accès et les modèles de contrôle de flux sont introduits en 1970, comme les premiers modèles de sécurité. On distingue les modèles de contrôle d'accès selon la façon dont est choisie la représentation de l'autorité d'administration. Lorsque celle-ci est répartie, on parle de modèle discrétionnaire DAC (Discretionary Access Control) [Lampson, 1971]. Dans le cas d'une administration centralisée, on parle de modèle à autorisation obligatoire ou de modèle mandataire MAC (Mandatory Access Control) [Bell & LaPadulla, 1976]. Il existe des variantes de ces politiques de sécurité telles que RBAC, IBAC, OrBAC ou ABAC.

Dans la suite, nous décrivons les principaux modèles afin de déterminer le plus pertinent pour notre système.

1.3.1.1 Politiques de sécurité discrétionnaires

Les modèles de politiques discrétionnaires (DAC - Discretionary Access Control) considèrent que chaque sujet peut détenir un droit de possession sur un objet. Le propriétaire de l'objet peut alors accorder des droits sur son objet à d'autres sujets. Il en résulte cependant un problème de perte de confidentialité de l'information.

Plusieurs modèles sont associés au DAC : le modèle de Lampson, le modèle HRU, le modèle Take-Grant, le modèle TAM, etc.

1.3.1.2 Politiques de sécurité obligatoire

Contrairement aux modèles discrétionnaires (DAC), les modèles d'autorisation obligatoire (MAC -Mandatory Access Control) centralisent complètement l'autorité d'administration. Il s'agit d'une restriction des politiques de sécurité où les sujets ne peuvent altérer l'accès aux objets. Ainsi, le problème de perte de confidentialité décrit plus haut ne peut exister.

Les modèles associés au MAC : modèle de Bell-LaPadula, modèle de Biba, modèle de Clark et Wilson, modèle de muraille de Chine.

1.3.1.3 Contrôle d'accès basé sur l'identité

Le contrôle d'accès basé sur l'identité (IBAC - Identity Based Access Control) [Lampson, 1971] est parmi les premiers modèles de contrôle d'accès. Ce modèle introduit les concepts fondamentaux de sujet, d'action et d'objet décrit ci-dessus

L'objectif de ce modèle IBAC est de contrôler tout accès direct des sujets aux objets via l'utilisation des actions. Ce contrôle est basé sur l'identité du sujet et l'identificateur de l'objet, d'où le nom du modèle IBAC.

Dans IBAC, une permission a le format suivant : un sujet a la permission de réaliser une action sur un objet. La politique d'autorisation qui permet de spécifier les permissions est définie grâce à l'utilisation d'une matrice de contrôle d'accès dans laquelle les lignes et colonnes de la matrice correspondent respectivement à l'ensemble des sujets et des objets du système d'information.

	Objet 1	Objet 1	Objet 1
Sujet 1	rw	r	w
Sujet 2	r	-	rw
Sujet 3	w	-	r

Cependant la limite de ce modèle est sa mise à l'échelle. En effet, la politique devient complexe à maintenir lorsque le nombre d'entités est important.

Le modèle RBAC introduit une abstraction de l'entité sujet qui devient rôle, permettant ainsi de réduire cette complexité.

1.3.1.4 Contrôle d'accès basé sur les rôles

Le contrôle d'accès basé sur les rôles (RBAC- Role Based Access Control) [Sandhu & al, 1996] est un modèle de contrôle d'accès dans lequel les décisions d'accès sont basées sur les rôles auxquels l'utilisateur est attaché. Dans RBAC, la politique de contrôle d'accès ne s'applique pas directement aux utilisateurs comme dans les modèles de contrôle d'accès DAC, ou MAC, les permissions ne sont plus associées de manière directe aux sujets, mais par le biais de rôles, qui regroupent des sujets qui remplissent les mêmes fonctions. Un rôle découle généralement de la structure d'une entreprise. Les utilisateurs exerçant des fonctions similaires peuvent être regroupés sous le même rôle. Un rôle, déterminé par une autorité centrale, associe à un sujet des autorisations d'accès sur un ensemble d'objets.

La modification des contrôles d'accès n'est pas nécessaire chaque fois qu'une personne rejoint ou quitte une organisation. Par cette caractéristique, RBAC est considéré comme un système « idéal » pour les entreprises dont la fréquence de changement du personnel est élevée.

L'objectif de ce modèle est de permettre la structuration de l'expression de la politique d'autorisation autour du concept de rôle (un concept organisationnel) : des rôles sont affectés aux utilisateurs conformément à la fonction attribuée à ces utilisateurs dans l'organisation. Le principe de base du modèle RBAC est de considérer que les autorisations sont directement associées aux rôles. Dans RBAC, les rôles reçoivent donc des autorisations de réaliser des actions sur des objets. Comme IBAC, le modèle RBAC ne considère que des autorisations positives (permissions) et fait donc l'hypothèse que la politique est fermée. Un autre concept introduit par le modèle RBAC est celui de session. Pour pouvoir réaliser une action sur un objet, un utilisateur doit d'abord créer une session et, dans cette session, activer un rôle qui a reçu l'autorisation de réaliser cette action sur cet objet. Si un tel rôle existe et si cet utilisateur a été affecté à ce rôle, alors cet utilisateur aura la permission de réaliser cette action sur cet objet une fois ce rôle activé. Lorsqu'un nouveau sujet est créé dans le système d'information, il suffit d'affecter des rôles au sujet pour que ce sujet puisse accéder au système d'information conformément aux permissions accordées à cet ensemble de rôles.

1.3.1.5 Contrôle d'accès basé sur l'organisation

Les modèles cités (DAC, MAC, RBAC, IBAC, …) ne permettent de modéliser que des politiques de sécurité qui se limitent à des permissions statiques. Ils ne permettent pas la possibilité d'exprimer des règles contextuelles [Abou El Kalam & al, 2003].

Le contrôle d'accès basé sur l'organisation (OrBAC Organization Based Access Control) [Abou El Kalam & al, 2003] est un modèle qui permet de spécifier des politiques de sécurité contextuelles relatives aux permissions, interdictions, obligations et recommandations. C'est un modèle qui permet d'exprimer une politique de sécurité au niveau organisationnel, c'est-à-dire indépendamment de l'implantation qui sera ensuite faite de cette politique. Il est ainsi possible d'exprimer l'ensemble des exigences de sécurité du système d'information et ensuite de distribuer ces exigences sur les différents composants de sécurité, vus comme autant de sous-organisations de l'organisation que constitue le système d'information. Cette distribution porte également sur les responsabilités d'administration que l'on peut confier à des sujets affectés à des rôles différents.

OrBAC utilise la notion de hiérarchie de rôle c'est-à-dire un mécanisme d'héritage de permission à travers la hiérarchie de rôle. Ceci peut être applicable dans le cas d'une organisation ayant des sous organisations ; les organisations se succèdent de père en fils. Mais dans le cas des organisations évoluant indépendamment et se situant au même niveau de hiérarchie, on ne peut pas appliquer cette notion de hiérarchie de rôle d'OrBAC mais chercher un autre moyen pour faire une extension.

OrBAC fournit un cadre pour exprimer les politiques de sécurité d'organisations, cependant il ne répond pas aux besoins de distribution, de collaboration et d'interopérabilité entre organisations. La technologie des services Web est un exemple pour fournir quelques mécanismes en particulier pour la collaboration (d'après l'article « Approche pour le contrôle d'accès collaboratif dans les infrastructures critiques » de [Baina & al, 2007]).

Une autre approche a été développée dénommée polyOrbac. Cette approche se base sur deux composants : contrôle d'accès OrBAC et technologie des services web.

1.3.1.6 Contrôle d'accès basé sur les attributs

La simple mise en place d'une politique de sécurité ne suffit pas pour sécuriser l'inter-opération. Il faut également tenir compte de son évolution par rapport aux besoins des différents participants à l'inter-opération [Coma, 2009]. Ce qui amène à penser aux modèles de contrôles d'accès contextuels comme le contrôle d'accès basé sur les attributs (ABAC – Attribute Based

Access Control). Pour inter-opérer, il faut prendre en compte les informations environnementales.

Le modèle ABAC a été développé par Eric Yuan et Jin Tong [Yuan & Tong, 2005], dans le but de pallier aux difficultés que rencontrent les architectures web services en terme de sécurité. En effet, les accès à l'information au niveau de ces architectures web services se font non seulement sur les systèmes distribués mais très dynamiquement. Les modèles classiques sont généralement destinés à un fonctionnement statique, ils ne permettent guère une évolution dynamique. De part sa définition, le modèle ABAC peut être le plus adapté pour les architectures fonctionnant dans un environnement ouvert « in the cloud » où différentes organisations peuvent assurer à la fois les accès aux informations et la protection de leurs ressources.

Comme son nom l'indique, le modèle ABAC définit les autorisations d'accès en se basant sur des caractéristiques de chaque entité, appelés attributs. Trois groupes d'attributs se distinguent selon le type de l'entité à laquelle ils s'appliquent :

1. Les attributs des sujets : un sujet est une entité qui peut agir sur une ressource. A chaque sujet on associe des attributs qui définissent son identité et ses caractéristiques. Par exemple le rôle du sujet peut aussi être considéré comme un attribut, tout comme le nom, le prénom, ou le titre, etc.
2. Les attributs des ressources : C'est un objet du système sur lequel un sujet peut agir. Autrement dit, c'est une entité qui peut être accessible à un sujet. Une ressource peut être un fichier, un service, etc. A chaque ressource est associée des attributs, variables selon sa nature, mais qui peuvent être : son type, le nom de son auteur, son propriétaire, la date de modification, etc.
3. Les attributs d'environnement : l'environnement peut être décrit par des informations opérationnelles, techniques, liées à la situation ou encore au contexte dans lequel l'accès à l'information se produit. La particularité du modèle, ABAC est la prise en compte du contexte d'exécution du système, en définissant des attributs d'environnement, comme par exemple : la date, le niveau de sécurité du réseau, le débit de la connexion, etc.

Cette particularité de prise en compte du contexte est très important dans notre problématique où nous sommes appelé à gérer des tiers sujets n'appartenant à aucune organisation et qui souhaitent obtenir un service auprès de tierces organisations.

Un des avantages de ce modèle est d'exploiter les opportunités offertes par les autres modèles, par exemple avec l'attribut du sujet « rôle » on peut appliquer le modèle RBAC.

Formalisme de politique de sécurité ABAC :

- S, R et E sont respectivement les sujets, les ressources et les environnements ;
- SA_k, ($1 \leq k \leq K$), RA_m, ($1 \leq m \leq M$), AE_n ($1 \leq n \leq N$) sont respectivement les (k-ième, m-ième et n-ième) attributs d'un sujet, d'une ressource, d'un environnement (avec k, m et n compris entre 1 et le nombre d'attribut définit pour chaque entité) ;
- ATTR(s), ATTR(r) et ATTR(e) sont les relations d'attributions des attributs aux entités (sujet, ressource et environnement) respectivement.

$ATTR(s) \subseteq SA_1 \times SA_2 \times \ldots \times SA_k$
$ATTR(r) \subseteq SA_1 \times SA_2 \times \ldots \times Sa_{2m}$
$ATTR(e) \subseteq SA_1 \times SA_2 \times \ldots \times SA_{2n}$

Les prédicats d'attributs sont définis de la façon suivante :

- *« CurrentDate(e)* = 6/12/2008 » signifie qu'on affecte la valeur 6/12/2008 à l'attribut d'environnement *CurrentDate* ;
- « Role(*s*) = "Service Consumer" » le sujet *s* joue le rôle de consommateur de service
- *« ServiceOwner*(*r*) = "XYZ, Inc." » la resource *ServiceOwner* est libélée par XYZ,Inc.

Les règles sont définies comme étant des fonctions booléennes des attributs de *s*, *r* et *e*.

On définit ensuite une politique comme un ensemble de règles regroupant plusieurs sujets et plusieurs ressources au sein d'un même domaine de sécurité. La gestion des autorisations se fait alors via l'évaluation de l'ensemble des règles de la politique.

$$Rule\ (X) : can\_access\ (s, r, e) \leftarrow f(ATTR(s), ATTR(r), ATTR(e))$$

Soit un ensemble de sujets et de ressources. On définit pour chaque sujet un attribut nommé «rôle», et pour chaque ressource un attribut « Name ». La règle « Les managers peuvent accéder aux ressources nommées ApprovePurchase » s'exprime alors de la façon suivante :

$$Rule\ 1 : can\_access\ (s, r, e) \leftarrow (Role(s) = 'Manager') \wedge (Name(r) = 'ApprovePurchase')$$

Avantages du modèle ABAC [Yuan & Tong, 2005] :

- Introduction de la notion de gestion de contexte via les entités d'environnement. Ceci permet d'obtenir des modèles plus souples, qui peuvent s'adapter à différentes situations et de dynamiser ainsi la prise de décision pour le contrôle d'accès. Cela est très appréciable pour l'application sur des SOA (Services Orienté Architectures), ou de nombreux paramètres peuvent influer sur la disponibilité de certaines ressources.
- Un modèle beaucoup plus adapté à la principale problématique liée au contrôle d'accès au sein de SOA, qui était le dynamisme d'accès aux informations.
- L'utilisation des attributs offre une granularité très fine pour définir les règles

d'autorisation : Il suffit de définir un attribut pour prendre en compte un nouveau paramètre entrant en jeu dans la définition des autorisations, qu'il s'applique aux sujets, aux ressources ou aux environnements.

ABAC [Yuan & Tong, 2005] se base sur des identités et des attributs qu'il considère comme acquis et sûrs. La mise en place d'une politique de sécurité suivant ce modèle implique donc la mise en place d'autres mécanismes pour assurer :

- La gestion des identités ;
- Leur certification ;
- La déclaration des attributs ;
- Le positionnement de ces derniers pour chaque entité du système.

Tous ces mécanismes [Yuan & Tong, 2005] devront mettre en jeu des mécanismes cryptographiques afin d'assurer l'intégrité des données échangées, et donc leur authenticité si les émetteurs sont certifiés.

Notons également que le modèle ABAC ne gère aucun mécanisme de délégation qui permettrait à un utilisateur de transférer ses droits à un autre. Ce principe, même s'il correspond souvent à une attente réelle de la part des organisations mettant en place des politiques de sécurité, est rarement modélisé, de part la complexité des mécanismes mis en œuvre.

Positionnement par rapport à nos travaux :

- Par rapport à la mobilité, ABAC correspond à notre approche car il présente un modèle dynamique basé sur les attributs.
- Il définit un formalisme basé sur les attributs qui offre plus de flexibilité en termes de spécifications des conditions de contrôle d'accès.
- Une architecture qui s'adapte à l'architecture orienté service et la technologie web service.
- La simplicité des règles de contrôle d'accès d'ABAC offre un avantage en interopérabilité mais l'administration devient complexe.

1.3.2 Les modèles de sécurité collaboratifs

La collaboration est le partage, l'échange des données, des services et des ressources entre différents organisations interconnectées et interdépendantes. Elle nécessite donc une architecture décentralisée qui fait appel à la notion d'interopérabilité.

Les modèles de contrôle d'accès collaboratifs peuvent être basés sur un mode de gestion de politique de sécurité centralisée ou décentralisée.

1.3.2.1 Système de gestion de politique de sécurité centralisé

Dans cette approche [Baina, 2009], les décisions de contrôle d'accès sont mises en œuvre dans un point central appelé « super organisation » qui impose sa politique de sécurité à toutes les autres organisations. Cette technique consiste à intégrer des composantes de la politique de sécurité globale dans chacune des politiques de sécurité des organisations. La politique de sécurité globale est applicable sur tout l'environnement collaboratif et est gérée par la « super Organisation », autorité reconnue par l'ensemble des organisations. L'avantage de cette méthode est la mise en place d'une plate-forme unifiée pour appliquer différentes politiques de sécurité au sein d'un seul système central.

Quelques travaux ont eu lieu sur cette approche. [Capitani & Samarati, 1996] définissent dans leurs travaux un contrôle d'accès pour les systèmes fédérés (Acces Control in Federed Systems) qui est un système de gestion de politique de sécurité globale et centralisée. Bertino et Jajodia ont mis en place une architecture de sécurité XACML basée sur cette approche [Bertino & al, 1996], [Jajodia & al, 2001].

1.3.2.2 Système de gestion décentralisée de sécurité (contrôle d'accès pair à pair)

Chaque organisation est responsable de ses propres décisions de contrôle d'accès et de leurs mises en œuvre. Le principe consiste à regrouper ou fusionner les politiques de sécurité des différentes organisations en une politique globale unique [Cuppens & al, 2006].

Le processus de regroupement de politique de sécurité utilise la notion de médiateur qui sert le point de liaison de politiques [Wiederhold, 1992].

D'autres travaux [Shehab & al, 2005] ont présenté une plate-forme distribuée sans médiateur; les décisions de contrôle d'accès étant prises indépendamment dans chaque domaine.

[Lorch & al, 2003], dans leur article « First Experiences Using XACML for Access Control in Distributed Systems », ont présenté le langage de contrôle d'accès XACML comme une plate-forme d'autorisation distribué et interopérable.

[Sturm & al, 2008], dans leur article « Mécanisme de contrôle d'accès des machines participantes », ont montré comment les politiques en XACML exportées par les participants du réseau pair à pair peuvent être combinées. Cette combinaison se fait par l'établissement d'une correspondance entre les politiques exportées : l'accès par l'un des participants est conditionné par un contrat établi entre les deux parties.

1.3.2.3 Exemple des modèles collaboratifs

1.3.2.3.1 Le modèle Multi-OrBAC

Multi-OrBAC [Abou El Kalam & Descarte, 2006] est une extension du OrBAC pour les systèmes multi organisationnels complexes, hétérogènes, interopérables et distribués. C'est un modèle de sécurité dynamique et adaptable, permettant d'un côté de spécifier des politiques de sécurité différentes dans chaque organisation, et de l'autre d'imposer des règles pour les interactions entre organisation.

Ainsi, pour décrire les primitives OrBAC dans la multi organisation, l'utilisation de Multi OrBAC permet de faire une extension. Cette extension est la modification des concepts de rôles, vues, et activités. Multi OrBAC a introduit une nouvelle notion rôle dans l'organisation qui est simplement une extension de rôle dans OrBAC. Il en est de même pour les vues et activités qui sont des extensions respectivement des vues et activités dans OrBAC.

1.3.2.3.2 Le modèle d'organisation virtuelle

Défini pour une durée limitée, ce modèle regroupe un ensemble d'organisations qui unissent leurs compétences et ressources pour répondre à une opportunité qu'elles n'auraient pu prendre en charge seule. Une fois le service accompli, le regroupement est démantelé [Naser, 2006].

1.3.2.3.3 Le modèle Organisation to Organisation (O2O)

C'est une extension du modèle OrBAC basée sur deux concepts : Virtual Private Organisation et Role Single-Sign On. Ce modèle permet de gérer l'interopérabilité et la collaboration entre des entités ayant leurs propres politiques de sécurité définies dans différentes organisations [Cuppens & al, 2006] et [Coma, 2006].

1.3.2.3.4 Le modèle Poly-OrBAC

Ce modèle est également basé sur OrBAC et sur les interactions par services web. PolyOrBAC [Baina, 2009] est une plateforme de contrôle d'accès collaboratif basé sur OrBAC et la technologie web services. Cette plateforme est applicable dans le contexte d'une infrastructure critique en générale et plus particulièrement dans le cadre d'un réseau électrique.

1.4 Les systèmes de contrôle d'accès distribués

Nous avons situé notre cadre de travail dans un environnement ouvert. Dans un tel environnement, la politique de sécurité utilisée se base essentiellement sur un système de contrôle d'accès distribué.

Plusieurs solutions utilisent actuellement le principe de la fédération d'identité. Cette technologie permet à un utilisateur de s'authentifier auprès de son fournisseur d'identité (Identity Provider - IdP) et d'accéder aux autres systèmes du cercle de confiance. L'utilisateur est dans ce type d'architecture généralement connu d'un unique fournisseur d'identité, dit d'appartenance. Prouver cette appartenance à un fournisseur de service du cercle de confiance peut suffire à obtenir l'accès au service de ce dernier.

Dans l'environnement ouvert que nous considérons, les fournisseurs d'identité de l'utilisateur sont multiples. Toute entité possédant une information sur un sujet est potentiellement un fournisseur d'identité. De plus nous considérons le cas où pour donner un accès à l'utilisateur, il lui faut fournir des informations certifiées par de multiples fournisseurs d'identité. Dans ce cas, la fédération dans son exploitation simple qui vient d'être faite ne suffit pas.

1.4.1 Privilege and Role Management Infrastructure Standards Validation (PERMIS)

PERMIS (Privilege and Role Management Infrastructure Standards Validation) a été conçu par l'université de Kent dans le but de mettre en place un système de contrôle d'accès distribué basé sur les certificats. PERMIS implémente un mécanisme de contrôle d'accès basé sur le modèle RBAC et utilise les certificats d'attributs X.509. Les règles d'autorisation RBAC sont définies à l'aide du langage XML et stockées dans un annuaire LDAP [Wu & Periorellis, 2005].

1.4.2 eXtensible Access Control Markup Language (XACML)

XACML (eXtensible Access Control Markup Language) est une spécification normalisée par l'OASIS [OASIS, 2005]. Il s'agit d'un langage XML dédié au contrôle d'accès. Il inclut un langage protocolaire de type requêtes/réponses pour la prise de décision et un langage pour la définition de politiques de contrôle d'accès.

Les spécifications XACML décrivent l'architecture modulaire suivante :

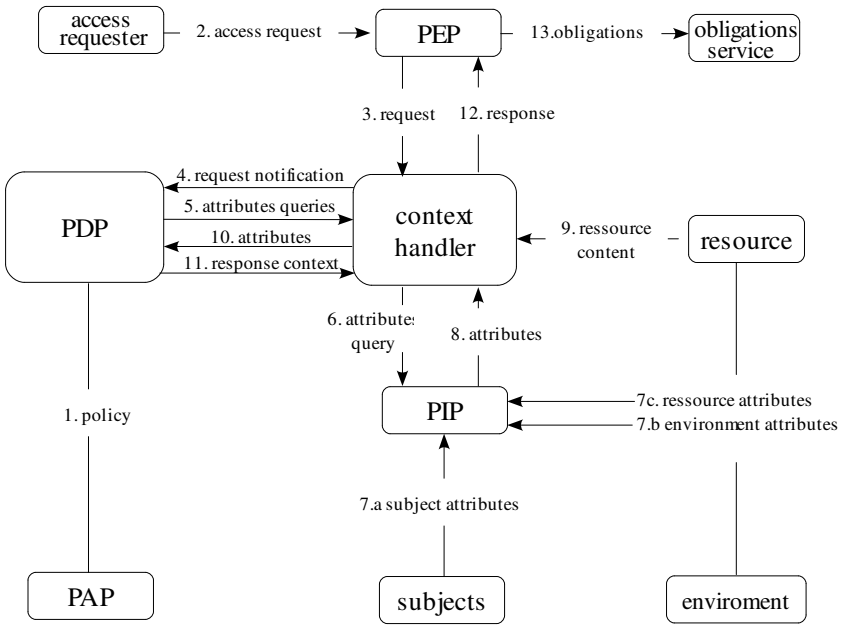

Figure 1.5 *: Architecture de gestion XACML [OASIS, 2005]*

1. PAP (Policy Administration Point) : Le point d'administration des politiques permet de définir les règlements sur lesquels vont s'appuyer les décisions prise par le PDP (PDP – Policy Decision Point).
2. L'utilisateur envoi sa requête au PEP (Policy Enforcement Point), point de mise en exécution des politiques, généralement implanté au sein du fournisseur de service.
3. Le PEP envoie la requête d'accès au « context handler » au format natif (langage supporté par le PEP), en y incluant optionnellement les attributs des sujets, des ressources, des actions et de l'environnement.
4. Le « context handler » construit un contexte de requête XACML et l'envoie au PDP.
5. Le PDP peut demander au « context handler » des attributs supplémentaires pour le sujet, la ressource, et l'environnement.
6. Le « context handler » demande ces attributs au point d'information (PIP – Policy Information Point).
7. Le PIP obtient les attributs demandés.
8. Le PIP retourne les attributs demandés au « context handler ».
9. Le « context handler » envoie les attributs demandés au PDP. Le PDP évalue la politique de contrôle d'accès.

10. Le PDP retourne le contexte de réponse XACML (incluant la décision d'autorisation) au « context handler » .
11. Le « context handler » traduit le contexte de réponse XACML au format de réponse du PEP. Le « context handler » retourne la réponse au PEP qui applique la décision d'autorisation.

XACML est un langage riche qui permet à la fois d'exprimer des politiques ABAC et RBAC.

Une fois les politiques définies, il est nécessaire d'expliciter la façon dont le système va effectuer les vérifications des règles avant de fournir ou non l'autorisation d'accès. L'architecture d'autorisation définit par ABAC est la suivante [Yuan & Tong, 2005]:

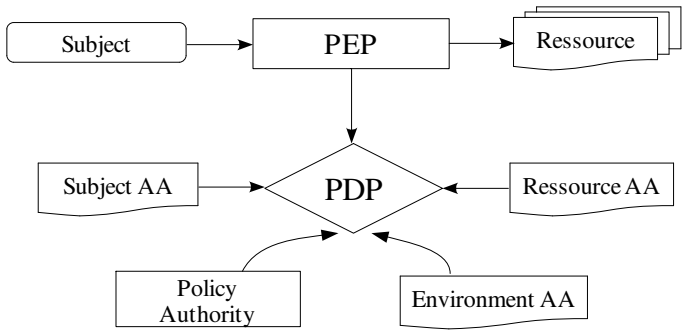

Figure 1.6 : Architecture d'autorisation ABAC (source [Yuan & Tong, 2005])

- Les AA (pour Atributes Authorities) sont les entités responsables de la création et de la gestion des attributs. Ils sont également responsables d'établir les relations entre les attributs, leur valeur et l'entité correspondante.
- Le PEP (pour Policy Enforcement Point) est l'entité chargée d'effectuer les requêtes d'autorisation, et d'appliquer la politique. Il est logiquement situé entre les sujets et les ressources, ce qui lui permet d'intercepter toute tentative d'accès, d'effectuer la requête vers le système de sécurité afin de vérifier si la tentative d'accès est autorisée ou non.
- Le PDP (pour Policy Decision Point) est l'entité chargée d'évaluer les politiques applicables et de prendre une décision concernant une requête d'accès à une ressource par un sujet. Il reçoit donc les requêtes du PEP, contacte les différents AA pour récupérer les attributs qui ne sont pas présents dans la requête, et applique les règles de sécurité pour donner sa décision au PEP (accès autorisé ou refusé).
- La Policy Authority crée et gère les politiques de contrôle d'accès (règles de décision, conditions, etc).

1.5 Analyse sur le contrôle d'accès et la confiance

Dans notre étude, nous souhaitons qu'un utilisateur inconnu du fournisseur de service puisse obtenir un accès au service au travers de la confiance du fournisseur de service envers des organisations attestant des informations sur l'utilisateur. Le sujet n'appartient pas à une unique organisation mère comme c'est le cas dans de nombreux travaux. Nous supposons qu'il est connu de multiples organisations et qu'il peut donc obtenir des certificats de ces multiples organisations afin d'obtenir des droits auprès d'organisations tierces. En effet, il est courant de voir des travaux sur l'interopérabilité où il est fait la supposition qu'il suffit à l'utilisateur d'arriver dans une organisation partenaire et de présenter un unique certificat, contenant potentiellement des attributs.

Permettre un accès suppose que l'on détienne au minimum une information sur l'identité du demandeur. L'identité sera définie à partir des attributs d'identité. Il s'agit donc d'exploiter le modèle de contrôle d'accès basé sur les attributs (ABAC).

Le système que nous allons définir va permettre à des entités inconnues de revêtir des rôles dynamiquement au moment des requêtes d'accès. Les objets, les droits nécessaires pour y accéder, et les rôles, seront définis dans une politique de contrôle d'accès lors de la mise en œuvre du système. En cours, d'exploitation les utilisateurs apportent des informations d'identité certifiées qui permettent au fournisseur de services d'enrôler dynamiquement les utilisateurs. Ces informations sont issues d'organisations tierces. L'organisation qui base son contrôle d'accès sur des informations provenant d'organisations tierces fait donc confiance à ces organisations. Ce qui justifie qu'un tel contrôle d'accès soit appelé « Trust management », [Blaze & al, 1996]. Pour reconnaître qu'une information provient bien d'un tiers de confiance, une organisation peut établir une connexion sécurisée avec celui-ci et l'authentifier (puis obtenir les informations nécessaires). Il est également possible que le tiers de confiance signe de l'information. Cela permet l'élaboration de documents (Certificats, assertions, security tokens, etc.) qui peuvent permettent d'utiliser ces informations sans nécessiter une connexion directe entre les deux organisations concernées.

Si l'on souhaite voir un tel contrôle d'accès déployé de manière globale, il est nécessaire de supposer une architecture de confiance globale où de nombreuses organisations se font confiance (par exemple au travers de partenariats commerciaux).

Lorsque l'accès est offert à une entité externe, il est possible de lui créer une identité au sein de l'organisation hébergeant le fournisseur de service. Ainsi, il est possible d'offrir ce service à cette entité sans qu'elle n'ait d'autres informations à fournir que de prouver une identité. Ce n'est cependant pas obligatoire, il est possible de fournir des accès à des entités pour une unique

transaction. Les informations employées pour le contrôle d'accès sont dans ce cas à présenter à chaque requête d'accès au service.

1.6 Conclusion

Nous avons présenté la notion de l'organisation virtuelle dans laquelle un certain nombre d'organisations peuvent collaborer et échanger des informations. La mise en place d'un tel système nécessite l'établissement de la confiance entre les différentes entités.

Nous utilisons dans notre système le modèle de contrôle d'accès ABAC qui s'adapte bien dans la mise en œuvre d'un système de contrôle d'accès basé sur la confiance en environnement ouvert. Ce modèle s'adapte également à la gestion dynamique des autorisations basées sur les rôles.

Nous utiliserons le langage XACML comme langage d'expression de la politique de contrôle d'accès. C'est un langage qui permet la définition de règles basées sur les attributs.

CHAPITRE 2

NOTRE SYSTEME

DE CONTROLE D'ACCES

2 NOTRE SYSTEME DE CONTRÔLE D'ACCES

2.1 Introduction

Dans un environnement ouvert tel que l'Internet, une requête provenant d'un sujet quelconque n'est pas soumise à un identifiant unique auquel serait associé une permission. Le contrôle d'accès en environnement ouvert ne peut s'appuyer que sur des tiers de confiance. Tiers de confiance à même de certifier des données concernant le sujet et qui permettront au fournisseur de service de déterminer si l'accès est autorisé.

Nous considérons que ces données portant sur un sujet sont des attributs d'identité. Le modèle de contrôle d'accès ABAC tel que défini dans la littérature par Eric Yuan [Eric & al, 2005] est un choix de modèle naturel. Il est employé de manière à intégrer dans les conditions la notion de sources de confiance des attributs d'identité.

Le contrôle d'accès se base donc sur les attributs d'identité émis par des sources de confiance du fournisseur de service. La signature numérique sur les données que sont les attributs d'identité permet au fournisseur de service d'authentifier la source de confiance [Saadi, 2009]. Les certificats comme vecteur de cette confiance, les formats et protocoles de diffusion de ces certificats sont étudiés au chapitre suivant.

Nous nous intéressons dans ce chapitre au contrôle d'accès par le fournisseur de service. Nous traitons donc du modèle de contrôle d'accès, du règlement envoyé à l'utilisateur, de l'analyse du règlement par l'utilisateur pour déterminer les sources de données pertinentes, la collecte par l'utilisateur des certificats d'attributs, la diffusion au fournisseur de service de ces certificats et la validation du règlement de contrôle d'accès à partir des certificats.

Voici le séquencement des opérations lorsque l'utilisateur demande un accès auprès du fournisseur de service. Le fournisseur de service possède une politique de contrôle d'accès au service exprimant les règles définissant les conditions à satisfaire. Les conditions pour obtenir l'accès requis sont extraites et rassemblées au sein d'un règlement que le fournisseur de service délivre à l'utilisateur. L'utilisateur détermine à partir de ce règlement quelles sont les sources d'attributs d'identité qu'il sera nécessaire de solliciter pour obtenir les données qui pourront satisfaire le règlement. L'utilisateur recueille les certificats et les diffuse au fournisseur de service. Le fournisseur détermine à partir des certificats si l'utilisateur est autorisé à accéder au service.

Ce chapitre présente un algorithme de traitement des règlements. Cet algorithme permet en fonction d'une règle et d'un ensemble d'attributs d'identités de déterminer si la règle est satisfaite. Cet algorithme est employé par le fournisseur de service pour autoriser l'accès en fonction des certificats reçus de l'utilisateur. L'utilisateur emploie également cet algorithme avant l'envoi des certificats pour déterminer si la diffusion peut résulter en une autorisation d'accès.

L'utilisateur va également employer une version simplifiée de cet algorithme afin de déterminer les sources d'attributs d'identité à solliciter. En effet, nous considérons que les certificats utilisés sont à durée de vie courte et ne sont donc pas stockés par l'utilisateur. Ils sont par contre délivrés à la demande de l'utilisateur. L'utilisateur possède un descripteur des attributs d'identité pour chacune des sources d'attributs, aussi appelées fichier de métadonnées des sources d'attributs. Ainsi, la version simplifiée de l'algorithme s'appuie sur les descripteurs des sources pour déterminer quelles sont les sources à solliciter. A ce stade, l'utilisateur ne dispose pas des valeurs des attributs et ne peut donc pas vérifier si le règlement peut être satisfait. Par contre, il peut déterminer s'il peut *a minima* obtenir les attributs impliquées dans les conditions du règlement. De cet algorithme résulte la liste des sources à solliciter, et pour chacune, les attributs à requérir. Suite à la récupération des certificats, l'utilisateur peut tester si le règlement est satisfait.

Ce fonctionnement est résumé par la figure 2.1 ci dessous.

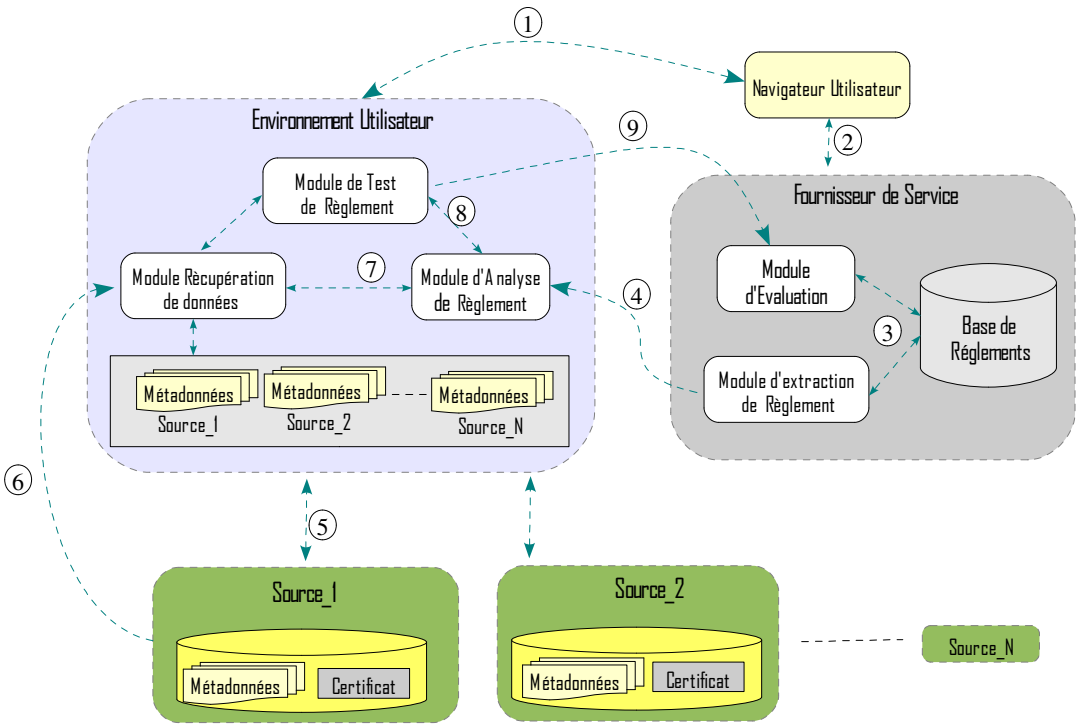

Figure 2.1 : Description de fonctionnement du système

1. Authentification de l'utilisateur auprès de son environnement (EU)
2. Demande de service auprès du fournisseur de services (SP)
3. Extraction du règlement
4. Envoi du règlement à l'utilisateur pour analyse
5. Recherche des informations auprès des sources
6. Récupération des données
7. Analyse des données
8. Test de règlement
9. Envoi des certificats au SP pour une prise de décision

2.2 Processus de traitement d'une requête d'accès

Nous décrivons à la figure 2.2 le processus de traitement d'une requête d'accès sans distinguer la répartition des fonctions entre le fournisseur de service et l'environnement de l'utilisateur.

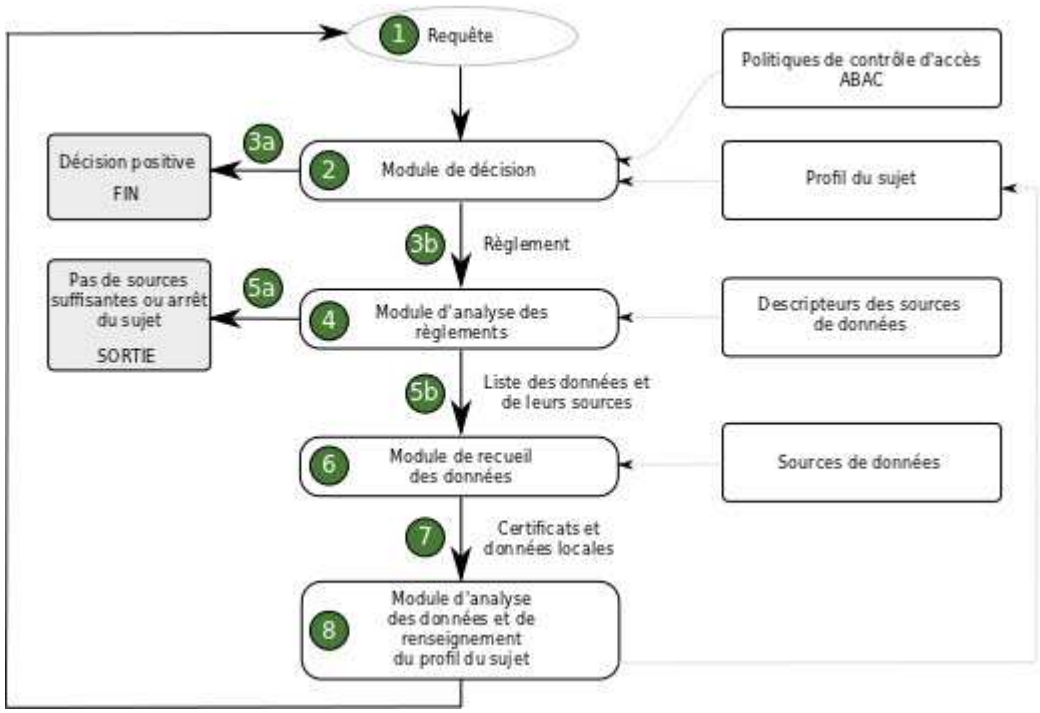

Figure 2.2 : Différentes étapes du traitement côté SP

1. Le traitement débute par une requête d'accès d'un sujet sur un objet pour mener une action.

Les règlements de contrôle d'accès de type ABAC sont constitués de règles du type :

permission (objet, action, règle). Les règles sont des expressions logiques du premier ordre sur des prédicats portant sur les attributs.

2. Lorsque le module de décision reçoit la requête, il détermine dans les politiques de

contrôle d'accès quelles sont les permissions qui s'appliquent à cette requête et les évalue.
Pour cela, le système de décision maintien pour chacun des sujets un profil du sujet contenant l'ensemble de ses informations d'identités, contenues dans des registres locaux ou obtenues au travers de certificats issus de tiers de confiance.
Pour permettre une nouvelle demande de certificats auprès du sujet en cas de décision négative le système de contrôle d'accès est à maintien d'état. Ainsi le module de décision n'est pas à maintien d'état mais le profil constitue l'état du système.
Cela permet de mettre en œuvre un procédé incrémental de recueil des certificats.
3.a. Si une permission est satisfaite, l'accès est autorisé, le traitement se termine.
3.b. Sinon, il est extrait de chaque expression les conditions non satisfaites. Une règle est générée à partir des conditions non satisfaites.
4. Le module d'analyse des règlements reçoit une telle règle. Ce module dispose pour ce sujet de l'ensemble des descripteurs des sources de données.
Les descripteurs des sources indiquent pour chaque source quelles informations peuvent être obtenues.
Ce module peut alors déterminer si une expression peut être satisfaite ou non en fonction des sources connues.
Il peut également définir quelles sont les informations d'identité minimales à obtenir pour satisfaire le règlement.
Ce module aura également en charge les interactions avec le sujet pour déterminer s'il veut ou non poursuivre la transaction, faire des choix dans les sources d'informations possibles, etc.
5.a Si la règle ne peut être satisfaite ou le sujet ne souhaite pas procéder à l'analyse du règlement, le traitement s'arrête.
5.b Sinon, le module délivre un descripteur des données à rassembler auprès des sources.
6. Le module de recueil des données, est en charge de rassembler ces données (il implémente les protocoles, par exemple LDAP, SAML, etc.). Il n'est pas obligatoire que l'ensemble des données soient recueillies. Le profil et la génération de règle permettent cela.
7. Un ensemble de certificats et données locales est envoyé au module d'analyse des données et de renseignement du profil du sujet.
8. Le module d'analyse des données et de renseignement du profil du sujet valide les certificats, en extrait l'information utile et l'injecte dans le profil du sujet. Le traitement reprend ensuite du début.

2.3 Attributs, espaces de noms et interopérabilité

Nous considérons que tous les acteurs de l'environnement possèdent une sémantique commune. En d'autres terme, il n'y a pas d'incertitude quant au sens des attributs d'identité.

Il peut y avoir plusieurs espaces de nommage des attributs, cependant, nous considérons qu'il est possible de faire un « mapping » sans incertitude entre espaces de nommages. Cela nous permet de supposer que nous travaillons avec un unique espace de nommage des attributs. Cette hypothèse est accréditée par de nombreux déploiements avec succès d'architecture distribuées, notamment sur le Web. Par exemple, les opérations financières et bancaires à l'échelle mondiale sont possibles grâce à une sémantique et un espace de nommage commun des données.

Ainsi, les attributs portent le même nom au sein du règlement de contrôle d'accès d'un fournisseur de service d'une organisation A et au sein du descripteur d'une source d'attributs d'une organisation B.

Nous décrivons dans la suite de cette section une liste des attributs d'identité qui sont principalement utilisés dans le contrôle d'accès basé sur les attributs d'identité :

Type de certificat	**Autorité**	**Type de certificats**	**Identité civile**	**Forme actuelle – envisagée**
Carte d'identité	Capable de vérifier l'identité civile	eID	Source	CNIE
Carte d'étudiant	Organisme accrédité par le ministère de l'enseignement supérieur et de la recherche	studentID	Associée	carte d'étudiant
Diplôme	Organisme accrédité par le ministère de l'enseignement supérieur et de la recherche	Diploma, engineerDiploma	Associée	Diplôme d'ingénieur, etc.
Permis de conduire	Préfecture	DrivingLicense, airLicence	Associée	Permis de conduire
Facture	Commerçant	bill	Associée	Factures utilisée pour du crédit d'impôts
E-Cash/jetons	Banques – Producteurs de "valeurs" pour un tiers	Money, Ecash, ePayment	Normalement non – éventuellement pour des besoins de lutte contre la double dépenses	Carte de crédit – Paypal
Accréditation/Procuration (roles – droits sur des objets délivrés pour un tiers)	Banques – Producteurs de "valeurs" pour un tiers	?	Associée	
Justificatif de domicile	Capable de vérifier un domicile et d'y associer une identité civile	?	Associée	Facture EDF – FT
Attestation d'assurance	Assurance	AttestRespCivil	Associée	

Tableau 2.1 : Récapitulatif des informations

2.4 Exemple d'un règlement de contrôle d'accès basé sur les attributs

Le contexte dans lequel nous travaillons nécessite l'utilisation d'un langage riche en expressivité pour exprimer nos conditions.

Pour une première illustration du besoin, notre choix s'est porté sur le langage d'expression CARL [Camenisch, 2010]. CARL est un langage conçu pour exprimer des contraintes fortes sur la diffusion d'information d'identité. Nous souhaitons donc illustrer notre cas d'usage avec ce langage.

Prenons l'exemple du cas d'usage de la location de voiture. Le fournisseur de service de location s'appuie sur un règlement de contrôle d'accès pour définir les conditions d'accès à une location.

Nous avons ajouté les primitives suivantes au langage :

- `pseudonym.isRevocable(Etat, cdts)` signifie que l'État fournit un pseudonyme. Pseudonyme qui peut être révoqué pour révéler l'identité réelle de la personne sous certaines conditions.
- `AssocCyberNotaire+` définit un ensemble de sources de confiance. La confiance dans ces sources est établie au travers d'une autorité racine, ici l'autorité racine des « cyber notaires ».
- `blind(adresse).isRevocable(AssocCyberNotaire+, cdts)` Signifie que l'association de CyberNotaire fournit une adresse mais de façon inintelligible. Celle-ci pourra être rendu intelligible sous certaine conditions.

Dans le règlement pris en exemple nous avons noté :

- `r` pour désigner le sujet
- `CNIE` pour Carte Nationale d'Identité Électronique

Règlement :

```
access: location_voiture
from: loueur_xy
to: any(r) where r = CNIE.pseudonym
in:
concat(r.CNIE.nom,r.CNIE.prenom) == concat(r.justificatif_domicile.nom,r.justificatif_domicile.prenom) ET
concat(r.CNIE.nom,r.CNIE.prenom) == concat(r.permis_conduire.nom,r.permis_conduire.prenom) ET
concat(r.CNIE.nom,r.CNIE.prenom) == concat(r.AttestRespCivil.nom,r.AttestRespCivil.prenom) ET
r.CNIE.pseudonym.isRevocable(Etat,cdts) ET
r.CNIE.age > 18 ET
{EDF,GDF,TelcoFilaire+}.{r.justificatif_domicile }.blind(adresse).isRevocable(AssocCyberNotaire+,cdts) ET
r.permis_conduire.validité > aujourdhui() ET
r.permis_conduire.points > 6 ET
r.AttestRespCivil.{Assoc_assurance+}.certificat_assuré.validité > aujourdhui() ET
{VISA,Paypal}.{eCash,epaiment}.argent == montant(transaction)
```

Ce règlement premet d'exprimer les conditions suivantes :

- Les valeurs des attributs « nom » et « prénom » dans les certificats doivent être égales.
- Il faut être âgé de plus de 18 ans.
- Il est nécessaire d'avoir un permis de conduire en cours de validité et disposant de plus de 6 points.
- Il faut être détenteur d'une assurance valide.
- Il faut disposer du montant de la transaction.

2.5 Modèle de données

Nous définissons dans cette section le modèle de données nécessaire à l'expression des règles ABAC et qui est ensuite utilisé par nos algorithmes.

Une règle de contrôle d'accès ABAC contient trois éléments principaux :

- L'expression logique combinant des prédicats ;
- Les prédicats portant sur un ou plusieurs attributs ;
- La description pour un attribut d'un nom, d'un espace de nom, et éventuellement d'une valeur et d'une liste de sources de confiance.

Nous définissons chaque structure de données à l'aide du concept de classe ce qui nous permet de définir le diagramme suivant :

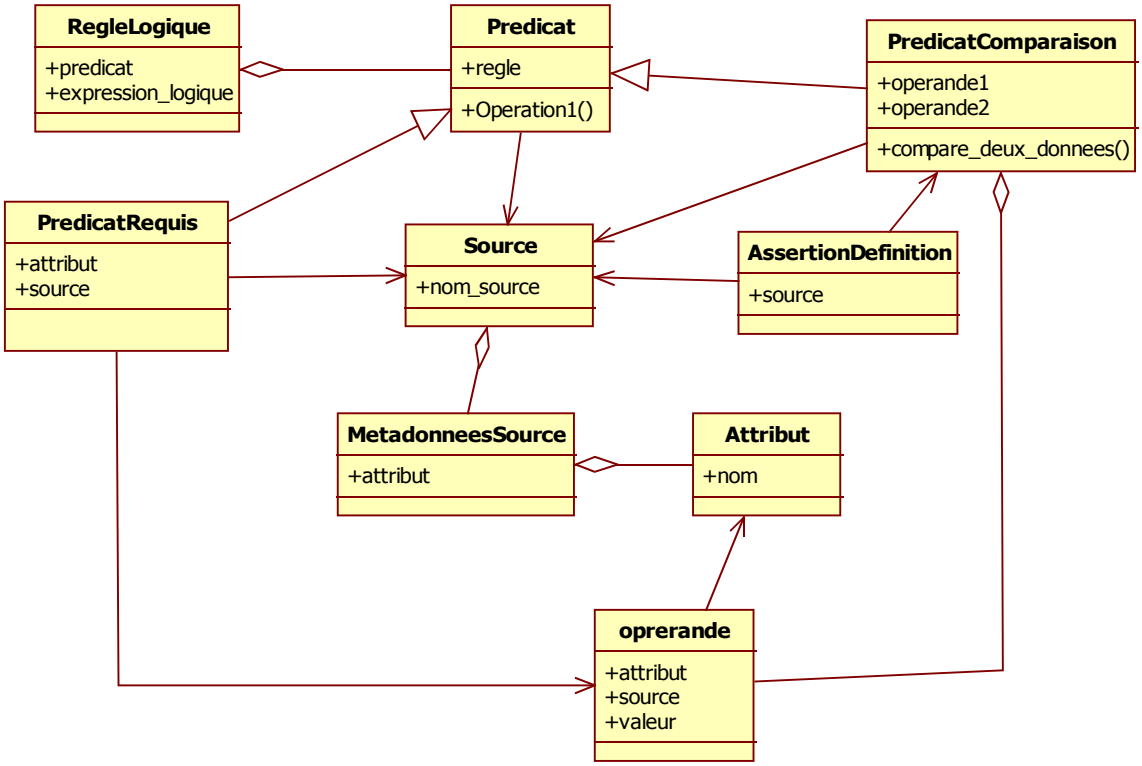

Figure 2.3 : Diagramme des classes

2.5.1 Définition d'attribut

Nous choisissons de définir chaque attribut. Ainsi, un prédicat portera sur une définition d'attribut et une information extraite d'un certificat sera reconnue si elle respecte une définition d'attribut connue du système.

Nous choisissons également de faire l'association entre plusieurs espaces de nom dans la définition d'attributs. Ceci a l'intérêt de pouvoir définir des règles sur un attribut quel que soit l'espace de nom employé par le fournisseur d'attributs. Cela impose de maintenir les associations des dénominations d'attributs entre espaces de noms.

Une définition d'attribut porte le nom donné à cet attribut dans notre système auquel est ajouté un élément d'association de noms par espace de noms. Nous nommons cet entité *AttributeDefinition*.

Une définition d'attribut possède également le type de données de l'attribut. Les types employés sont ceux indiqués dans la spécifications XACML3 core Annexe 1.2 [OASIS, 2010]. Par exemple : 'http://www.w3.org/2001/XMLSchema#string' ou 'urn:oasis:names:tc:xacml:2.0:data-type:ipAddress'

Afin de décrire un attribut avec ses valeurs une entité AttributeData est définie. Elle associe un tuple de valeurs à une définition.

2.5.2 Source de confiance

Une source de confiance désigne soit un tiers de confiance, soit un ensemble de tiers de confiance en décrivant une autorité racine [Cofta, 2007].

Une source de confiance est une entité possédant les trois attributs : nom, clé publique ou certificat et type.

Un tiers de confiance désigné directement sera une source de confiance de type "DIRECT'. S'il s'agit d'une autorité de confiance racine, la source de confiance sera de type "ANCHOR'. Une source peut être également locale à l'hôte, une telle source sera de type "LOCAL". Enfin, une source de type "SELF" sera déclarée au sein du système pour désigner les attributs qui ne sont pas issus d'une source de confiance mais par exemple déclarés sur l'honneur par l'utilisateur.

Pour rendre la notion d'autorité de confiance, nous nous restreignons à la notion d'architecture à clé publique X509 [Rivest, 1978]. Ainsi, on trouvera dans le champ certificat d'une source de confiance de type "ANCHOR" le certificat X509 de l'autorité de certification racine désignée. Cette source de confiance permet de désigner comme tiers de confiance, toutes les sources qui signeront leurs certificats d'attributs avec une clé publique contenue dans un certificat à clé publique X509, lui-même signé par une autorité de certification X509 qui est dans le chemin de confiance ayant pour racine l'autorité de certification racine désigné par la source.

2.5.3 Assertions

Nous introduisons un type d'éléments appelés *Assertion* qui permet de lier une définition d'attributs, ou des valeurs d'attributs, à une source.

Une assertion de définition, appelée *AssertionDefinition*, est utilisée dans les prédicats décrits par la suite. Il s'agit d'indiquer que l'on souhaite qu'un attribut proviennent d'une ou plusieurs sources, et éventuellement d'imposer une contrainte sur sa ou ses valeurs. Nous introduisons donc un élément *AttachedSource* qui permet de déclarer plusieurs sources attachées à une *AssertionDefinition*.

Une assertion de données, appelée *AssertionData*, est utilisée dans deux cas :

- Pour indiquer une valeur de comparaison dans un prédicat, auquel cas, il n'y a pas de source

attachée (les contraintes sur les sources étant défini au niveau de l'assertion de définition).

- Pour peupler un élément appelé profile lors de la réception de certificats d'attributs, auquel cas, l'assertion de données est attachée à une unique source.

Un prédicat peut porter sur deux opérandes qui sont, soient des assertions de définition, soient des assertions de données. Les classes *AssertionDefinition* et *AssertionData* dérivent d'une classe parente appelée *AssertionAny.*

2.5.4 Prédicats

Un prédicat exprime une contrainte sur un ou plusieurs attributs, ou leurs valeurs. La contrainte sur l'attribut peut également porter sur la source. L'évaluation d'un prédicat retourne un booléen.

Lorsqu'un prédicat porte sur un attribut, des objets de type *AttributeDefinition* sont employés. Lorsqu'un prédicat porte sur la ou les valeurs d'un attribut, des objets de type *AttributeData* sont employés.

Nous définissons une classe parente pour tous les prédicats appelée *Predicate*.

Le prédicat élémentaire vise à indiquer qu'il est souhaité qu'un attribut provenant d'une source soit fourni. Nous nommons ce prédicat *PredicateRequired*. Cette entité prend comme champ un objet de type *AssertionDefinition*.

Notons que n'importe quel autre type de prédicat implique que les attributs sur lesquels porte le prédicat sont *de facto* requis.

Les prédicats sont nombreux, nous citons donc ici les principes généraux de définition des prédicats et quelques exemples de la façon choisie pour les mettre en œuvre. L'annexe A.3 de XACML3 définit un très grand nombre de prédicats suffisant pour exprimer la très grande majorité des cas d'usage.

Le prédicat d'égalité sert à poser une contrainte sur l'égalité de valeur de deux attributs (nom de source A est égale à nom de source B) ou entre un attribut et une valeur déterminée (Age de source A est égale à 18), et cela quel que soit le type de données de l'attribut (chaîne, entier, date, etc.). A noter une exception, la comparaison de chaîne peut se faire en ignorant la casse.

Nous définissons donc une structure de données unique appelée prédicat de comparaison, soit *PredicateComparison*, qui s'applique de la même façon à n'importe quel type de données. Un

prédicat de comparaison possède un attribut *type* qui détermine le type de comparaison (égalité, infériorité ou égale, etc.). La fonction d'évaluation utilisera ce paramètre pour déterminer le traitement.

Les deux attributs comparés peuvent être définis à l'aide de deux définitions d'attribut différentes. Par exemple, pour comparer un nom de famille et un prénom. Par contre, les types de données des définitions doivent être les mêmes.

Un prédicat de comparaison a deux champs *operande1* et *operande2*.

Il existe plusieurs possibilités dans le traitement des attributs multi-évalués. Nous nous limitons aux choix suivants.

Pour l'égalité de deux attributs, le prédicat est vrai si :

- au moins une valeur de chaque attribut est égale ;
- les valeurs d'un attribut sont un sous-ensemble des valeurs de l'autre attribut ;
- les ensembles de chaque attribut sont identiques.

Pour la comparaison d'attributs, le prédicat est vrai si :

- au moins un couple de valeur (une valeur de chaque attribut) satisfait la contrainte ;
- toutes les valeurs de l'opérande 1 doivent satisfaire la contrainte avec toutes les valeurs de l'opérande 2 (le nombre de valeurs peut être différent);

Les comparaisons "inférieur ou égale" et "supérieur ou égale" nécessitent de déterminer un couple d'option.

2.5.5 Expression logique

Il s'agit de définir une expression logique du premier ordre combinant les prédicats ce qui constitue la règle à satisfaire pour obtenir une permission.

Une expression logique comporte cinq éléments de syntaxe : '&', '|', '-', '(' et ')'

'&', '|', '-' signifient 'ET', 'OU', 'NON' et sont identifiés en XACML par urn:oasis:names:tc:xacml:1.0:function:and, urn:oasis:names:tc:xacml:1.0:function:or , urn:oasis:names:tc:xacml:1.0:function:not.

Une expression logique peut être notée sous la forme d'une chaîne, par exemple :

(p1 | ((p2 & p3) | (-p4)))

Nous définissons une entité *AbacRule* avec un champs *expression* qui est une chaîne représentant l'expression logique. Cette chaîne contient les éléments de syntaxe et les prédicats sont remplacés par les identifiants des instances des prédicats. Cela implique que chaque instance de prédicat possède un identifiant unique qui ne contient pas les éléments de syntaxe de l'expression logique.

2.5.6 Constitution d'un règlement

Les permissions sont des entités possédant trois champs : un objet, une action et une règle ABAC portant sur les attributs d'un sujet unique.

L'objet peut être un conteneur, appelé vue, d'objet ou d'autres conteneurs d'objets.

L'action peut être un conteneur, appelé activité, d'action ou d'autres conteneurs d'actions.

Considérons un règlement comme la définition d'un ensemble de permissions. Des sous-ensembles peuvent être définis pour déterminer par exemple à quelles applications ils s'appliquent. Il est ainsi possible de définir des espaces de noms pour chaque application ce qui autorise des identifiants d'objet redondants entre applications.

Nous considérons que chaque politique possède un unique espace de nom (qui n'a rien à voir avec les espaces de nom des attributs). L'appartenance d'une entité à un espace de nom définit l'appartenance à une politique. L'ensemble des entités dans un espace de nom définissent une politique.

2.5.7 Métadonnées de sources

Un descripteur de source, qui permet à l'utilisateur de savoir les attributs d'identités que fournit une source, est décrit par un objet *DescripteurSource* qui contient un ensemble de définitions d'attributs.

2.6 Algorithmes de traitement

Nous nous intéressons aux éléments algorithmiques suivants :

- La vérification d'une règle dans un règlement en fonction de valeurs d'attributs. Cet algorithme est utilisé par le fournisseur de service à la réception des certificats de l'utilisateur et par l'utilisateur avant l'envoi des certificats.
- La vérification de la disponibilité, auprès des sources, des attributs impliqués dans une règle grâce aux métadonnées des sources. Cet algorithme est utilisé par l'utilisateur à la réception

du règlement pour déjà savoir s'il peut obtenir les attributs nécessaires, et ensuite, pour savoir quelles sont les sources à solliciter. Cet algorithme est une simplification du précédent algorithme puisque les valeurs d'attributs ne sont pas testées.

- L'évaluation de l'expression logique est un algorithme utilisé par ces deux algorithmes.

2.6.1 Analyse des sources pertinentes par le module d'analyse des règlements (Algorithme A1)

Cette analyse permet à l'utilisateur de déterminer si les informations demandées par le fournisseur de service sont disponibles dans les sources. Il n'y a pas de vérification sur les valeurs des données, autrement dit la satisfaction des conditions n'est pas vérifiée.

Le module d'analyse des règlements extrait du règlement chaque définition d'attribut et détermine quelles sont les définitions qui se trouvent dans les descripteurs de sources.

Chaque prédicat, portant sur une définition présente dans les sources, est un atome Vrai de l'expression logique, Faux pour les définitions non trouvées.

La table de vérité de l'expression logique est ensuite établie.

Si aucune conclusion n'est vraie, le traitement s'arrête. Si plusieurs conclusions sont vraies, les combinaisons de prédicats sont classées par ordre croissant du nombre de prédicats remplacés par des atomes vrais. Cela permet de déterminer la diffusion minimale d'information d'identité satisfaisant le règlement de contrôle d'accès. Ce module a alors la charge d'interagir avec l'utilisateur pour déterminer le choix de l'utilisateur. Ainsi, il pourra préférer une source à une autre. Il pourra choisir de diffuser plus d'informations s'il sait qu'une diffusion inférieure pourrait conduire à un rejet, par exemple parce qu'il sait que son age n'est pas suffisant.

La fonction prend en entrée: le règlement et les métadonnées

- Les métadonnées sont des documents qui décrivent le profil de l'utilisateur dans les sources.
- Le règlement est constituée d'un ensemble de prédicats (P1, P2, …, Pn) et d'opérateurs logiques (op1, op2, …, opn).

  Ainsi une règle peut être représentée ainsi :

```
P1 op1 P2 op2 ... Pn
```

Prenons l'exemple suivant:

```
(Nom [S1, S2]) ET (prénom [S1]==prénom [S2]) ET (age [S2] >=18)
```

(Nom [S1, S2]) est un prédicat de type "requis" (*PredicateRequired*) qui indique une liste de

sources acceptée pour fournir l'attribut *Nom*.

(prénom [S1]==prénom [S2]) est un prédicat de type "comparaison" (*PredicateRequired*) qui indique que le *prénom* obtenu de la source *S1* doit être égal au *prénom* obtenu de la source *S2*.

`(age [S2] >=18)` est un prédicat de type "comparaison" (*PredicateRequired*) qui indique que l' *age* obtenu de la source *S2* doit être supérieur ou égale à la valeur *18*.

Voici la représentation graphique des éléments impliqués :

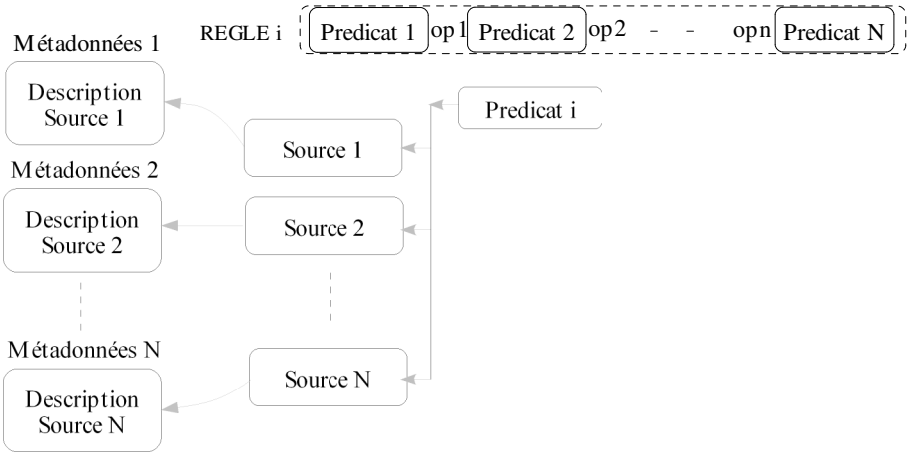

Figure 2.4 : Modèle de données d'une règle

L'algorithme parcourt tous les prédicats de la règle. Et pour chaque prédicat, l'algorithme doit être capable de dire si les informations recherchées sont disponibles dans les sources. Si tel est le cas, un vecteur de valuation permet d'indiquer la disponibilité de l'information demandée.

Deux cas se présentent selon que le prédicat est de type 'requis' ou de type 'comparaison'.

- Cas 1: Si le prédicat est de type « requis »:

  Dans ce cas deux possibilités se présentent encore : soit le prédicat indique une liste des sources (figure 2.4) dans lesquelles on doit chercher l'attribut, soit le prédicat n'indique rien.

  - Si le prédicat indique une liste des sources:

    Pour chaque source pointée, on vérifie si l'attribut désigné est disponible dans les métadonnées de la source.

  - Si le prédicat n'indique pas de source :

    N'importe qu'elle source fournissant cet attribut sera accepté. L'évaluation retourne toujours vrai puisque le fait de ne pas spécifier de source indique que l'utilisateur peut donner lui-même l'information, ce qui est équivalent à une source de type SELF.

- Cas 2: Si le prédicat est de type « comparaison »:

  Dans ce cas, le prédicat est constitué de deux opérandes (operande1 et operande2). Les opérandes qui portent sur une assertion de définition sont traités comme des prédicats « requis ». Les opérandes d'assertions de données sont ignorés.

Nous faisons le choix d'utiliser un langage informel pour exprimer nos algorithmes pour gagner en simplicité. Nous faisons également le choix d'exprimer les algorithmes dans une syntaxe proche du langage pour faciliter l'implémentation. A titre d'exemple, les blocs sont séparés par indentation.

Voici l'algorithme définit :

```
fonction recherche_disponibilite_informations_sources(rule, metadonnees_sources[]):

    valuation_predicats[]
    i=0

    Pour chaque predicat dans regle.predicats:

        valuation_predicats[i] = Faux

        Si predicat est de type 'Requis':
            # Le predicat indique une liste de sources
            Si predicat.sources non nul:
                Pour chaque source de predicat.sources et tant que valuation_predicats[i] == Faux:
                    Pour chaque attribut de metadonnees_sources[source].attributs et tant que valuation_predicats[i] ==
Faux:
                        Si predicat.attribut.nom est égale attribut.nom:
                            valuation_predicats[i] = Vrai
            # Le predicat n'indique pas une liste de sources
            Sinon:
                valuation_predicats[i] = Vrai

        Si predicat est de type 'Comparaison':
            operande1_trouve = Faux
            operande2_trouve = Faux
            Si predicat.operand1.valeur existe:
                # operande "valeur"
                operande1_trouve = Vrai
            Sinon:
                # operand qui repose sur un attribut d'une source
                Si predicat.operand1.sources non nul:
                    Pour chaque source de predicat.operand1.sources et tant que operande1_trouve == Faux:
                        Pour chaque attribut de metadonnees_sources[source].attributs et tant que operande1_trouve ==
Faux:
                            Si predicat.operand1.attribut.nom est égale attribut.nom:
                                operande1_trouve = Vrai
                Sinon:
                    operande1_trouve = Vrai

            Si predicat.operand2.valeur existe:
                operande1_trouve = Vrai
            Sinon:
                Si predicat.operand2.sources non nul:
                    Pour chaque source de predicat.operand2.sources et tant que operande2_trouve == Faux:
                        Pour chaque attribut de metadonnees_sources[source].attributs et tant que operande2_trouve ==
Faux:
                            Si predicat.operand2.attribut.nom est égale attribut.nom:
                                operande2_trouve = Vrai
```

```
            Sinon:
                operande2_trouve = Vrai

        Si operande1_trouve == Vrai ET operande2_trouve == Vrai:
            valuation_predicats[i] = Vrai

    i = i + 1
    retourne valuation_predicats
```

Une version plus riche de cet algorithme permet de retourner les sources pour lesquelles les attributs sont disponibles. Pour cela, il suffit que le vecteur de valuation ne contiennent pas de booléens mais une valeur nulle ou une liste. La liste indique les sources pour lesquelles l'attribut est disponible.

Pour résumer, cet algorithme permet à l'utilisateur de vérifier, à partir du règlement envoyé par le fournisseur de service, si les attributs sur lesquels portent les conditions sont disponibles auprès des sources connues de l'utilisateur.

2.6.2 Algorithme de vérification des prédicats d'une règle logique portant sur des attributs d'identité (Algorithme A2)

Le premier algorithme porte sur la vérification de la disponibilité des informations dans les sources. L'algorithme défini dans cette section possède une structure similaire. Simplement, au lieu de se réduire à la vérification de la présence d'un attribut dans un document de métadonnées d'une source, il vérifie que les valeurs des attributs satisfassent les prédicats.

Si le prédicat est de type « requis », la présence de l'attribut et la contrainte de source sont vérifiés.

Si le prédicat est de type « comparaison », la présence de l'attribut et la contrainte de source sont vérifiées pour chaque opérande de type *AssertionDefinition*. La comparaison est ensuite évaluée.

Nous supposons un objet appelé *UserProfile* dans lequel sont stockés tous les attributs obtenus des sources, ou pour le fournisseur, délivrés par l'utilisateur.

L'algorithme défini est le suivant :

```
fonction verification_des_predicats(rule, user_profile):

    valuation_predicats[i] = Faux

    Pour chaque predicat dans regle.predicats:

        Si predicat est de type 'Requis':
            Si existe attribut dans user_profile.attributes où attribut.nom == predicat.attribut.nom et si predicats.sources
où attribut.source dans predicat.attribut.sources :
```

```
                valuation_predicats[i] = Vrai

        Si predicat est de type 'Comparaison':
            valeur_operand1 = Nul
            Si predicat.operand1 est de type 'Requis':
                Si existe attribut dans user_profile.attributes où attribut.nom == predicat.operand1.attribut.nom et si
predicats.operand1.sources où attribut.source dans predicat.operand1.attribut.sources :
                    valeurs_operand1 = attribut.valeurs
            Sinon
                valeurs_operand1 = predicate.operand1.valeurs
            Si predicat.operand2 est de type 'Requis':
                Si existe attribut dans user_profile.attributes où attribut.nom == predicat.operand2.attribut.nom et si
predicats.operand2.sources où attribut.source dans predicat.operand2.attribut.sources :
                    valeurs_operand2 = attribut.valeurs
            Sinon
                valeurs_operand2 = predicate.operand2.valeurs
            Si valeurs_operand1 et valeurs_operand2:
                valuation_predicats[i] = evaluer_comparaison(valeurs_operand1, valeurs_operand2, predicat.type)

        i = i + 1

    retourne valuation_predicats
```

2.6.3 Algorithme d'évaluation des expressions logiques (Algorithme A3)

L'algorithme permet l'évaluation d'une expression logique. C'est une fonction ayant comme paramètres d'entrée une règle sous la forme d'une chaîne et la valuation des prédicats. En sortie, l'algorithme renvoie une liste de liste des prédicats par ordre croissant satisfaisant l'expression logique.

Dans le cas où l'évaluation n'est pas satisfaite, l'algorithme retourne une liste de liste de prédicats par ordre croissant du nombre de prédicats à modifier pour obtenir la satisfaction de l'expression logique.

```
fonction evaluation_constant(X):
    Si X est égale à 'Vrai':
        retourner Vrai
    retourner Faux

fonction evaluation_variable(X, Env):
    retourner Env[X]

fonction evaluation_atom(X, Env):
    Si est_de_type_constant(X):
        retourner evaluation_constant(X)
    else:
        retourner evaluation_variable(X, Env)

fonction evaluation(X, Env):
    Si est_de_type_atom(X):
        retourner evaluation_atom(X, Env)
    Si X[0] est égale à '(':
        n = 1
        Pour i dans l'intervalle(1, len(X)):
            Si X[i] est égale à '(':
                n = n + 1
```

```
            Si X[i] est égale à ')':
                n = n - 1
                Si n est égale à 0:
                    retourner evaluation(\
                        chaine(evaluation(X[1:i], Env)) + X[i+1:],
                        Env)
        retourner Faux

    Si X[0] est égale à '(' ET X[len(X)-1] est égale à ')':
        retourner evaluation(X[1:len(X)-1], Env)

    Si X[0] est égale à '-' ET est_de_type_proposition(X[1:]):
        Si evaluation(X[1:], Env) est Faux:
            retourner Vrai
        Si evaluation(X[1:], Env) est Vrai:
            retourner Faux

    Pour i dans l'intervalle(0, len(X)):
        Si X[i] est égale à '&' ET est_de_type_proposition(X[0:i]) ET est_de_type_proposition(X[i+1:]):
            Si evaluation(X[0:i], Env) ET evaluation(X[i+1:], Env):
                retourner Vrai
            retourner Faux

    Pour i dans l'intervalle(0, len(X)):
        Si X[i] est égale à '|' ET est_de_type_proposition(X[0:i]) ET est_de_type_proposition(X[i+1:]):
            Si evaluation(X[0:i], Env) or evaluation(X[i+1:], Env):
                retourner Vrai
            retourner Faux
```

La valuation d'une expression peut être de type atomique ou de type proposition.

- Si la valuation est de type atomique alors l'évaluation de l'expression logique peut porter sur une constante ou une variable.
  - L'évaluation de la constante est vraie alors si la règle est vraie.
  - L'évaluation de la variable est fonction de la valuation du prédicat dans la règle.
- Si la valuation est de type proposition alors on procède à une évaluation de façon récursive de l'expression logique.

Si l'évaluation de l'expression est fausse, on détermine l'ensemble des prédicats à modifier pour satisfaire l'expression. Ainsi on extrait toutes les variables dans la règle pour créer une liste des variables. Pour chaque élément de la liste des variables, on évalue l'expression. Si l'expression n'est pas satisfaite on retient l'élément. Cette évaluation se fait de façon récursive.

Prenons l'exemple d'une règle définie par : rule = (1&2&3) | ((4&5) & (-6)) et soit la liste des valuations des prédicats définie par valuation_predicat = {'1':False, '2':True, '3':False, '4':True, '5':False, '6':True}

L'évaluation de cette expression est fausse. L'algorithme retourne « False » ainsi qu'une liste de listes des prédicats à satisfaire, ordonné par taille croissante des listes.

Nous prenons l'exemple des deux premières listes retournées.

- Première liste : [('5', False, True), ('6', True, False)]
  - Il convient pour satisfaire la règle que la variable 5, passe de faux à vrai, et que la variable 6, passe de vrai à faux.
- Seconde liste : [('1', False, True), ('3', False, True), ('6', False, True)]
  - Il convient pour satisfaire la règle que les variables 1 et 3, passent de faux à vrai, et que la variable 6, passe de vrai à faux.

2.7 Conclusion

Nous avons décrit notre système de contrôle d'accès basé sur les attributs. L'expression de la politique de contrôle d'accès est basée sur le traitement de règlement ABAC. Le traitement se fait sur l'analyse des certificats.

Par exemple pour une location d'un véhicule, notre système apporte une amélioration considérable par rapport à d'autres méthodes existantes où le client doit être enregistré préalablement auprès de fournisseur de service. Pour cette location, il faut que le client soit enregistré préalablement auprès de services de location. Dans le système que nous concevons, pour obtenir un service, l'utilisateur collecte ses certificats et les diffuse au fournisseur de services. Les autorités de certification se portent garantes des informations délivrées.

Nous avons dans ce chapitre définit la problématique et un moyen de le résoudre, la diffusion d'un règlement et son analyse par l'utilisateur, la collecte et l'envoi des certificats. Nous avons déterminé le modèle de données et les algorithmes de traitement. Nous avons évalué théoriquement ces algorithmes.

Lorsque l'utilisateur exprime sa requête au fournisseur de service, ce dernier lui envoie un règlement à analyser et à satisfaire. Cette analyse ne peut se faire que grâce à un environnement utilisateur riche.

Cet environnement lui permet aussi de récupérer les certificats pour les fournir au fournisseur de service, ou pour certains certificats, fournir des autorisations au fournisseur de service pour que celui-ci les récupère directement. Cet environnement sert à l'utilisateur d'outil de gestion de ses certificats. Il s'agit donc d'un composant de l'architecture qui s'ajoute aux composants fournisseurs de service et source d'attributs d'identités.

Dans le chapitre qui suit nous présentons l'architecture protocolaire permettant de mettre en œuvre un environnement utilisateur riche dédié aux opérations présentées dans ce chapitre.

CHAPITRE 3

ARCHITECTURE

3 ACHITECTURE

3.1 Introduction

Dans ce chapitre nous traitons de l'architecture protocolaire du système de contrôle d'accès basé sur la confiance. Plus précisément nous nous focalisons sur les protocoles permettant d'établir une session auprès d'un système, sur les protocoles permettant de véhiculer un règlement de contrôle d'accès et sur les protocoles permettant d'obtenir puis de diffuser des informations d'identité entre tiers de confiance.

Il existe de nombreux standards couvrant les besoins de ces divers protocoles. Notre première contribution est la constitution d'un assemblage utilisant de multiples langages et protocoles standards permettant de constituer la majeur partie du système présenté au précédent chapitre. Notre seconde contribution est la présentation de protocoles permettant de compléter ce premier ensemble afin de parvenir au fonctionnement du système.

Le fonctionnement décrit au précédent chapitre repose sur une architecture 3-tiers de gestion d'identité dans laquelle le sujet est à même de recevoir les règlements de contrôle d'accès, de les analyser, de collecter les certificats et de répondre au fournisseur de service qui envoie le règlement.

Nous nous appuierons donc sur une architecture dans laquelle l'utilisateur dispose d'un environnement lui permettant de prendre part aux échanges protocolaires afin de pouvoir accomplir ces tâches. Nous faisons le pari qu'un tel environnement doit être en ligne afin de permettre la mobilité du terminal de l'usager, et non pas déployer sur les terminaux des usagers. Nous supposons cependant que tous les terminaux sont munis d'un client applicatif standard, soit un navigateur Web, permettant de consommer les services.

Le patchwork protocolaire intègre donc dans l'architecture un environnement avec lequel l'utilisateur pourra interagir au travers d'une interface graphique Web, environnement potentiellement hébergé par une organisation tierce du fournisseur d'identité et des organisations des sources de certificats.

Nous repoussons à plus tard la discussion sur les hôtes potentiels d'un tel environnement. Notons simplement qu'il pourrait s'agir d'un service en ligne potentiellement rémunérateur pour l'hébergeur, un dispositif multimédia comme la « freedom box » ou d'un téléphone personnel.

3.2 Les bases de l'architecture

3.2.1 Architecture de confiance

La donnée signée est le média de la confiance dans la mesure où la signature apporte l'authentification du signataire, tiers potentiel de confiance. La signature numérique permet de garantir l'authentification de l'origine d'un document électronique et son intégrité. Le principe de la signature numérique d'un document est de chiffrer une empreinte [Nissenbaum, 1998] de celui-ci avec la clé privée de son auteur. La vérification de la signature se fait avec la clef publique du signataire.

La publication de la clé publique est une problématique intimement liée à celle de la confiance. D'une part, des clés publiques vont déterminer les signataires de certificats de données et qui sont considérés comme des tiers de confiance pour la fourniture de ces données. D'autre part, des clés publiques vont déterminer des tiers de confiance pour la distribution de clés publiques. Leur exploitation, leur nombre et les protocoles vont dépendre du modèle de confiance utilisé. Nous avons vu au premier chapitre les divers modèles.

Une fédération est un partenariat entre un ensemble de partenaires. Le fonctionnement général d'une fédération d'identité [Pfitzmann & Waider, 2004] est la délivrance de certificats des organisations à leurs utilisateurs pour leur permettre d'accéder aux services des autres partenaires de la fédération. Ce mécanisme repose sur la confiance des fournisseurs de services envers les sources de certificats de partenaires.

Nous considérons dans notre environnement qu'un ensemble de liens de confiance existent entre les fournisseurs de services et les sources de données d'identité, donc que les fournisseurs de services sont à même d'authentifier les signatures réalisées par les sources qui leur sont de confiance.

En effet, dans le système décrit, l'utilisateur emploie des certificats issus de multiples sources afin d'obtenir un service. Cela suppose donc qu'il existe une architecture de confiance permettant aux fournisseurs de services d'établir des liens de confiance envers les sources d'attributs d'identité qu'elles jugent pertinentes.

Pour cela, chaque organisation peut avoir à sa charge d'établir une relation de confiance avec chacun de ses partenaires. Il est également possible qu'il existe des autorités de confiance, par exemple une autorité représentant une fédération, afin d'établir des liens de confiance avec l'ensemble des membres de cette fédération.

3.2.2 Session, Web SSO et identité certifié

Un mécanisme d'authentification permet d'apporter la preuve de connaissance d'un secret partagé avec un système [Grob, 2003]. Lorsque qu'un mécanisme d'authentification est associé à la notion d'identité et de compte, cela permet à un utilisateur d'apporter la preuve de possession d'une identité ou d'un compte. Il en découle généralement une ouverture de droit au sein du système permettant à l'utilisateur de faire l'usage de services sous l'identité prouvée. Nous appelons parfois l'authentification auprès d'un système, l'établissement d'identité ou l'établissement de session.

La notion d'authentification implique que l'identité de l'utilisateur est préalablement connu du système. L'utilisateur a par exemple un compte de messagerie et il s'authentifie pour établir une session permettant l'usage de sa messagerie. L'utilisateur peut être soumis à une interaction avec le système lors d'un procédé d'authentification, le plus courant étant la fourniture d'un mot de passe. Un système d'authentification unique (SSO pour Single Sign On) [Clerq, 2002] vise à réduire ces interactions pour améliorer l'ergonomie. Cela ne réduit en aucun cas le nombre d'établissement de session. Les mécanismes d'authentification sont modifiés pour éviter les interactions multiples de l'utilisateur. Un système de Web SSO est un tel système pour l'authentification de l'utilisateur auprès d'applications Web.

Le système OpenID [OpenID Fundation, 2012] a été conçu pour permettre aux usagers de services sur le Web de disposer d'un Web SSO personnel. L'identifiant OpenID est ainsi le moyen d'indiquer à un fournisseur de service l'adresse de son serveur de SSO personnel.

Le mot authentification est également employé lorsque l'usager peut obtenir une session auprès d'un système en apportant un identifiant signé par un tiers de confiance. Cela potentiellement sans avoir d'identité existante au sein du système. Le système autorise dans ce cas l'ouverture de session non pas parce qu'il reconnaît une identité mais parce qu'il obtient un identifiant correspondant à une identité qui est certifiée par un tiers de confiance. Il fait confiance à ce tiers pour connaître l'identité de l'utilisateur désigné par cet identifiant. Nous appelons ce mécanisme la certification d'une identité. Ce système est également couramment nommé délégation de l'authentification.

La technologie de fédération d'identité couple dans un même mécanisme le Web SSO [Andreas & Chris, 2003] (ou de simplification de l'établissement d'identité) et l'identité certifiée ce qui peut apporter une confusion sur ces deux fonctions distinctes.

La technologie SAML, qui constitue le standard de fédération d'identité, s’appuie sur la notion de fournisseur d'identité. Celui-ci délivre des certificats appelés assertions [Maler, 2006] qui

permettent à l'utilisateur de s'authentifier auprès de fournisseurs de services de partenaires. Ce certificat contient généralement un pseudonyme qui sert à la certification de l'identité. En outre, le fournisseur d'identité peut délivrer à l'utilisateur des certificats sans le ré-authentifier et destinés à de multiples tiers de confiance. Enfin, le pseudonyme peut être persistant pour un même couple utilisateur/service ce qui permet de l'utiliser comme moyen d'établissement d'identité, réalisant ainsi un système de Web SSO.

3.2.3 Attributs certifiés et non certifiés

Lors de la consommation d'un service auprès d'un fournisseur de service, il peut être nécessaire que l'utilisateur fasse parvenir divers attributs d'identité.

Lorsque ces données sont simplement déclaratives, elles peuvent servir à déclarer une adresse électronique ou une adresse de livraison par exemple. Il existe divers protocoles Web visant à faciliter la fourniture d'informations d'identité, généralement conçus dans le but d'éviter à l'utilisateur de les saisir de multiples fois.

Le protocole OpenID est aujourd'hui le protocole le plus abouti pour ce besoin. Un serveur OpenID constitue pour l'utilisateur un espace de stockage en ligne de ses attributs d'identité. Lorsque ce système est couplé à un système de Web SSO, le fournisseur de service peut demander des attributs en même temps que la demande d'authentification. Les attributs sont alors fournis en même temps que les données d'authentification.

Il existe des protocoles qui sont plus adaptés au partage de données qu'à l'échange d'attributs d'identité, mais qui peuvent compléter notre boîte à outil protocolaire. Prenons l'exemple du partage de données personnelles comme des photographies entre différents services. L'utilisateur peut souhaiter véhiculer des photographies d'un réseau social vers un fournisseur du service d'impression. Dans ce cas, le protocole OAUTH est le plus adéquat car il permet à l'utilisateur de choisir l'objet de l'échange selon le contexte (s'il veut faire imprimer une photographie, il va choisir une photographie). Chaque objet est ainsi identifié de manière unique pour chaque transaction. Ce protocole est également utilisé pour la délégation d'accès à un service. Ainsi, le protocole permet par exemple à l'utilisateur de donner l'autorisation à un tiers l'accès à un service personnel en son nom. Citons par exemple le service « twitter @anywhere », qui permet à un utilisateur de twitter à partir d'un site tiers.

Intéressons-nous maintenant aux protocoles de diffusion d'informations certifiées. Il s'agit toujours de véhiculer une donnée d'une source vers une destination, ou de déléguer un accès, et il

pourrait être légitime de penser que les mêmes protocoles peuvent être employés. Cependant ces deux usages ont fait diverger les protocoles. Les protocoles précédemment cités n'ont par exemple pas été nativement conçus pour intégrer les signatures, support de la confiance, ni les problématiques de respect de la vie privée.

De même, les usages font que les implémentations des précédents protocoles visent généralement à permettre à l'utilisateur de choisir librement les sources de ses attributs et données, c'est à dire sans que celles-ci soient préalablement connues du fournisseur de service, puisqu'un lien de confiance entre la source et la destination (mode déclaratif) n'est pas nécessaire. Au contraire, lorsque la source doit être de confiance, le fournisseur possède généralement une liste de tiers de confiance dans laquelle l'utilisateur choisit sa source.

Ainsi, nous avons introduit dans la section précédente le protocole SAML. Celui-ci permet également de véhiculer des attributs dans des certificats appelés assertions en réponse à une demande d'authentification ou en réponse à une requête d'attributs d'un fournisseur de service.

Un ensemble de protocoles s'appuyant sur SAML peut être employé pour mettre en œuvre des protocoles d'échanges plus complexes. Ce sont les spécifications IDWSF. Elles permettent ainsi aux différentes organisations de gérer des annuaires de services afin de faciliter leur découverte [Tourzan & Koga, 2007]. Ensuite, il est possible de permettre à l'utilisateur de délivrer des autorisations d'accès à ses services comme cela est fait avec OAUTH. Ainsi, il sera possible à une destination d'obtenir un jeton d'accès de la part de l'utilisateur afin d'obtenir des attributs d'identité certifiés ou d'accéder à un service.

3.2.4 Standard lié à l'autorisation

Divers efforts ont ainsi été conduits pour permettre la centralisation du service d'autorisation.

Le premier enjeu est celui de la centralisation de la définition d'une politique de contrôle d'accès. L'utilisation d'un outil unique permet de vérifier la cohérence de l'ensemble des règlements de contrôle d'accès avec un modèle. Il est ensuite possible d'en extraire des règlements en divers formats afin de les injecter dans les systèmes de contrôle d'accès des applications.

Il est ensuite également possible de centraliser la prise de décision. Ainsi, la logique d'interprétation des règlements est unique et est implémentée au sein d'un module de décision. Celui-ci est ensuite interrogé par les applications pour donner les décisions. Les applications n'ont plus qu'à exécuter les décisions.

Les spécifications XACML offrent un langage pour les règlements qui permet d'exprimer des règlements respectant les modèles RBAC ou ABAC.

Ce langage est parfois utilisé comme base de définition à partir de laquelle des règlements d'autres formats sont extraits. De la base de règlements XACML peuvent être ainsi extrait des règlements injectables dans diverses applications

Le module ne travaille cependant généralement pas directement à partir du format XACML. Celui-ci est généralement traduit sous la forme d'objets permettant des performances meilleures que l'analyse de documents XML.

XACML définit ensuite une architecture protocolaire permettant aux applications (appelées PEP) d'interroger un module de décision (appelé PDP).

Il s'agit d'un protocole de requête-réponse sous la forme de documents XML. Ces messages peuvent être échangés via le protocole SOAP sur HTTP [Lawrence, 2006] par exemple.

Un moyen élégant est également de coupler XACML et SAML pour permettre aux applications ayant établi une session pour un pseudonyme de l'utilisateur de demander une décision concernant cet identifiant. Ceci fait l'objet d'une spécification dédiée.

3.3 Les espaces de noms et la standardisation des documents

Nous faisons l'hypothèse qu'il n'y a qu'une ontologie décrivant notre système et que l'ensemble des acteurs s'accordent sur celle-ci. Chaque concept peut ainsi avoir un nom ou un identifiant dans plusieurs espaces de noms mais il est possible [Bray, 2006] d'établir une correspondance entre les différents identifiants d'un même attribut au sein de plusieurs espaces de nom.

Cette hypothèse est nécessaire pour espérer un système interopérable à grande échelle. C'est de fait une hypothèse valide au regard des architectures à grande échelle aujourd'hui en place.

Par exemple, supposons le concept « Nom de famille ». Notre supposition fait qu'il n'y a pas de confusion entre les intervenants du système sur le sens de l'attribut d'identité « Nom de famille ». On retrouve le « Nom de famille » dans plusieurs espaces de noms par exemple dans les spécifications LDAP (RFC4519), Identity Metasystem Interoperability (IMI) Version 1.0 http://docs.oasis-open.org/imi/identity/v1.0/identity.html (formerly Identity Selector Interoperability Profile (ISIP) - http://www.microsoft.com/download/en/details.aspx?displaylang=en&id=18221) . On peut ainsi faire la correspondance suivante :

Concept : Nom de famille

Espace de nom : 'RFC4519' (LDAP)
Chaîne identifiante : 'sn'
Alias de la chaîne identifiante : '2.5.4.4'

Espace de nom : 'http://schemas.xmlsoap.org/ws/2005/05/identity/claims' (IMI)
Chaîne identifiante : 'http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname'
Alias de la chaîne identifiante : 'Last Name'
Definition : (sn in RFC 2256) Surname or family name of a subject. According to RFC 2256: .This is the X.500 surname attribute which contains the family name of a person..

Ainsi, autant que possible nous utiliserons les standards suivants dans notre thèse :

- LDAP/X.500 pour les attributs d'identité.
- Les assertions SAML2, standard des certificats à faible durée de vie (http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf). L'exploitation des attributs LDAP/X.500 dans les assertions SAML étant spécifiée (http://docs.oasis-open.org/security/saml/SpecDrafts-Post2.0/sstc-saml-attribute-x500-cd-01.pdf).
- Pour les espaces de noms portant sur les concepts d'un règlement de contrôle d'accès : opérateurs logiques, concepts de rôles, permissions, règlements, nous nous appuierons intégralement sur la spécification cœur XACML v3.0 (urn:oasis:names:tc:xacml:3.0:core:schema:wd-17).

3.3.1 Standardisation des documents de règlements de contrôle d'accès

Nous formatons les règlements fournis à l'utilisateur à l'aide du langage XACML. Dans une architecture XACML centralisée, la politique de contrôle d'accès est centralisée au sein du point central de décision (PDP).

Notre travail consiste donc d'une part à exprimer les règlements ABAC qui puissent être interprétés par l'utilisateur comme un ensemble de certificats à présenter au fournisseur de services. Il s'agit d'autre part de pouvoir extraire un fragment de règlement en XACML correspondant à la requête de l'utilisateur.

Un règlement XACML débute toujours par une balise *Policy*. La balise contient ensuite plusieurs règles déclarées par un élément *Rule*. A titre d'illustration voici un document type avec une unique règle nommée « RentACar ». Une règle satisfaite donne une autorisation ou un refus

d'autorisation. Il s'agit de l'attribut E*ffect* qui peut prendre les valeurs « Permit » ou « Deny » :

```
<?xml version="1.0" encoding="UTF-8"?>
<Policy PolicyId="urn:ndg:security:1.0:authz:test:policy"
    xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:cd:04"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:policy:schema:cd:04 http://docs.oasis-open.org/xacml/access_control-
xacml-2.0-policy-schema-cd-04.xsd"
    RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-overrides">
    <Description>
        Dummy policy
    </Description>
    <Rule RuleId="RentACar" Effect="Permit">
        <!-- Regle a definir .
    </Rule>
</Policy>
```

La règle peut contenir divers éléments mais dans le cas présent nous nous contentons d'étudier l'élément *Condition* qui contient la règle logique et les prédicats exprimant les conditions à satisfaire.

Un élément *Condition* contient des éléments *Apply* qui ont chacun un type indiqué par l'attribut *FunctionId*. Tous les types sont définis à l'annexe A de la spécification coeur XACML oasis-access_control-xacml-2.0-core-spec-os .

Nous nous intéressons aux éléments logiques :

- urn:oasis:names:tc:xacml:1.0:function:and
- urn:oasis:names:tc:xacml:1.0:function:or
- urn:oasis:names:tc:xacml:1.0:function:not

Nous nous intéressons ensuite aux éléments qui permettent d'indiquer les comparaisons. Il en existe un nombre important et nous n'en citons que quelques uns à titre d'illustration :

- urn:oasis:names:tc:xacml:1.0:function:string-equal
- urn:oasis:names:tc:xacml:1.0:function:integer-equal
- urn:oasis:names:tc:xacml:1.0:function:integer-greater-than
- urn:oasis:names:tc:xacml:1.0:function:integer-greater-than-or-equal

A titre d'illustration nous créons une règle pour l'exemple suivant :

```
(age of (IdP1 or IdP2 or IdP3) >= 18 and surname of IdP1 == surname of IdP2
```

Ce qui donne :

```
<Condition>
    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
        <!--surname of IdP1 == surname of IdP2-->
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <SubjectAttributeDesignator
                AttributeId="urn:ndg:security:authz:1.0:surname"
                DataType="http://www.w3.org/2001/XMLSchema#string"
                Issuer="IdP1"/>
            <SubjectAttributeDesignator
                AttributeId="urn:ndg:security:authz:1.0:surname"
                DataType="http://www.w3.org/2001/XMLSchema#string"
                Issuer="IdP2"/>
        </Apply>
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:or">
            <!-- age of IdPx >= 18-->
            <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-greater-than-or-equal">
                <SubjectAttributeDesignator
                    AttributeId="age"
                    DataType="http://www.w3.org/2001/XMLSchema#integer"
                    Issuer="IdP1"/>
                <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">
                    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#integer">18</AttributeValue>
                </Apply>
            </Apply>
            <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-greater-than-or-equal">
                <SubjectAttributeDesignator
                    AttributeId="age"
                    DataType="http://www.w3.org/2001/XMLSchema#integer"
                    Issuer="IdP2"/>
                <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">
                    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#integer">18</AttributeValue>
                </Apply>
            </Apply>
            <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-greater-than-or-equal">
                <SubjectAttributeDesignator
                    AttributeId="age"
                    DataType="http://www.w3.org/2001/XMLSchema#integer"
                    Issuer="IdP3"/>
                <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">
                    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#integer">18</AttributeValue>
                </Apply>
            </Apply>
        </Apply>
    </Apply>
</Condition>
```

3.3.2 Métadonnées des sources

Nous utilisons JSON (JavaScript Object Notation) définit par [D. Crokford, 2006] pour la description des documents de métadonnées des sources d'attributs. Plusieurs schémas de métadonnées sont définis par des organismes de standardisation. Nous choisissons JSON pour la simple raison qu'il présente un format de données léger [D. Crokford, 2006], et simple à utiliser par rapport à d'autres types de schéma.

Le métadonnées des sources contient deux éléments principaux : la source et l'attribut (nom

de l'attribut et l'espace de nom). La source est représentée par un objet contenant les attributs : nom de source et le protocole utilisé pour l'échange.

La description en JSON est la suivante :

```
{
  "source":{
   "type" : "object" ,
    "properties": {
              "nom": {
                 "type": "string"
                  "description": "Nom de la source"
                 "required": true
                }
              "protocole" : {
                 "type" : "string"
                 "description": "protocole de communication"
                  "required": true
              }
  "Attribute": [
        {
           "AttributeName":{
                "type": "string"
                 "description": "Nom de l'attribut"
            },
         }
          "AttributeNamespace": {
               "type" : "string"
               "description": "Espace de nom defini pour l'attribut"
          }
     ]
}}}}}}}
```

Voici un exemple de documents de métadonnées des sources d'attributs pour une source qui fournit deux attributs dans l'espace de nom X500 et qui peuvent être obtenus via le protocole SAML2 :

```
{
"SourceName":"attributes_provider.xyz.com",
"Protocol":"urn:oasis:names:tc:SAML:2.0",
"Attributes":
        [
        {"sn","X500"},
        {"gn","X500"}
        ]
}
```

3.4 Conception de l'architecture

Nos travaux sur les protocoles se focalisent sur les éléments particuliers suivants de notre architecture :

- Un schéma global d'architecture avec les protocoles employés.

- Les authentifications multiples de l'utilisateur : il s'agit d'exploiter un Web SSO personnel.
- La déclaration des documents de métadonnées des sources d'attributs par l'utilisateur auprès de son environnement personnel : il s'agit d'un protocole que nous avons défini.
- L'envoi du règlement du fournisseur de service à l'environnement personnel de l'utilisateur : il s'agit d'un protocole que nous avons défini.
- L'envoi des certificats de l'environnement personnel de l'utilisateur au fournisseur de services : il s'agit d'un protocole que nous avons défini.

3.4.1 Schéma global de l'architecture

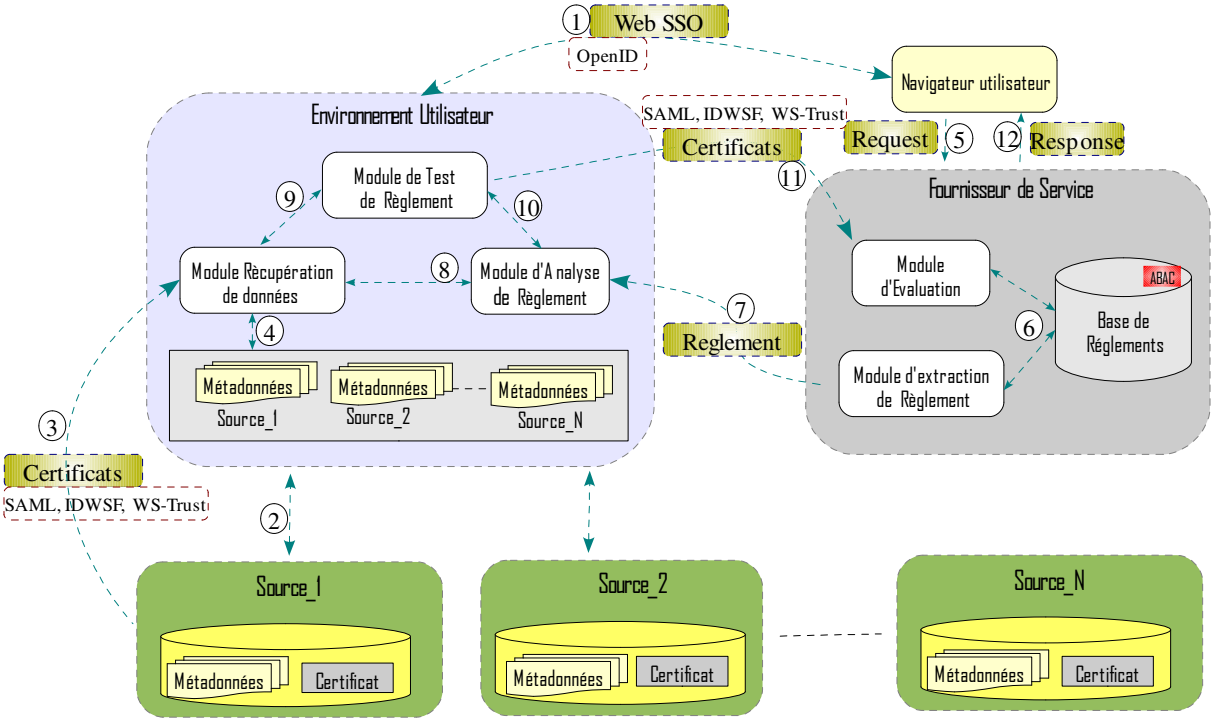

Figure 3.1 : Architecture protocolaire

Les différents échanges se résument sur ce schéma qui définit notre architecture :

Avant d'exprimer sa requête auprès du SP, l'utilisateur va d'abord récupérer ses informations personnelles auprès des différentes sources. Les phases 1, 2, 3 et 4 résument le processus de la récupération des métadonnées qui sont détaillées dans la section 3.4.3. Ensuite il va envoyer sa requête auprès du SP pour exprimer un besoin. Le SP reçoit sa requête, examine et extrait le règlement correspondant qui sera envoyé à l'utilisateur. L'utilisateur reçoit le règlement, l'analyse et présente au SP les certificats nécessaires à la prise de décision. Ce qui s'explique par les étapes suivantes:

1. L'utilisateur s'authentifie auprès de son environnement et demande les métadonnées;
2. Son environnement est capable de le rediriger vers les sources de confiance pour récupérer les métadonnées;
3. Les métadonnées sont récupérées grâce au module de récupération des données;
4. Les informations sont enregistrées dans les descripteurs des sources;
5. L'utilisateur exprime une demande auprès du SP;
6. Le SP examine sa requête et extrait le règlement correspond à la requête;
7. Le règlement est envoyé à l'environnement utilisateur. Il est analysé par le module d'analyse de règlement;
8. Pour analyser, ce module va recevoir les métadonnées qui seront envoyées par le module de récupération des données;
9. Le module de test reçoit le règlement pour une évaluation;
10. Le module de test va récupérer les certificats nécessaires et évalue le règlement;
11. A l'issue du test positif, les certificats sont envoyés au SP pour une prise de décision;
12. Après évaluation, le SP prendra une décision pour répondre à l'utilisateur.

Dans ce qui suit, nous montrerons quels sont les différents protocoles qui interviennent à chaque phase d'échange.

Pour la récupération des certificats, les protocoles utilisés peuvent être SAML ou WS-Trust [Lawrence, 2009].

3.4.2 Authentification et SSO

L'utilisateur doit s'authentifier auprès de son environnement utilisateur.

Puis, l'environnement utilisateur doit permettre à l'utilisateur de s'authentifier simplement auprès des fournisseurs d'identités et d'attributs pour ouvrir une session et obtenir les certificats.

L'utilisateur peut également posséder divers comptes auprès d'autres interlocuteurs et son environnement utilisateur doit lui permettre d'ouvrir une session auprès de ces divers environnements.

Pour cela, l'environnement utilisateur s'appuie sur un Web SSO. Pour une authentification avec redirection de l'utilisateur, nous avons choisis OpenID. Ce mécanisme servira également à

indiquer au fournisseur de service l'emplacement de l'environnement utilisateur lorsqu'il aura à rediriger l'utilisateur dessus.

Pour l'authentification auprès des fournisseurs de certificats, il pourra également être utilisé une authentification SSL avec une clé publique authentifiant l'environnement utilisateur. Ainsi l'environnement utilisateur pourra héberger un ensemble de clés publiques dédiées par le fournisseur d'attributs pour réaliser une partie du SSO [Suriadi & al, 2007].

La déclaration de la clé publique de l'environnement utilisateur auprès du fournisseur de certificats se fera lors d'une phase d'enregistrement qui permettra également à l'utilisateur d'injecter dans son environnement utilisateur le certificat d'un fournisseur de certificat ainsi qu'un point d'entrée applicatif permettant de récupérer le descriptif des informations personnelles pouvant être obtenues.

3.4.3 Déclaration de sources d'attributs auprès de l'environnent utilisateur

L'introduction entre l'environnement de l'utilisateur et un fournisseur de certificats se fait en trois phases pouvant être conduites indépendamment :

- La récupération du certificat d'authentification de la source;
- L'initialisation d'un moyen d'authentification de l'utilisateur auprès de la source ;
- La récupération du document de métadonnées.

  Nous nous intéressons principalement à la récupération du document de métadonnées.

Nous pouvons supposer que le fournisseur de certificats dispose d'une interface Web accessible au travers du protocole HTTPS. L'utilisateur peut ainsi vérifier avec son navigateur le certificat. Sur cette interface, l'utilisateur peut indiquer l'adresse de son environnement utilisateur. Le fournisseur de certificats redirige alors le navigateur de l'utilisateur vers son environnement personnel en donnant un nonce et deux URLs en paramètres de la requête HTTP GET. Le navigateur obtient du fournisseur de certificat un code de réponse HTTP 302 avec dans l'entête HTTP le champs *Location* suivant :

```
http://environnement_utilisateur.org/declaration_source/initiation?
nonce=_080667A86E27D0B1EF50E29852B2E2EF&url_disco=http://idp.idp.com/discovery&url_return=http://idp.idp.com/discov
eryReturn
```

L'utilisateur s'authentifie alors auprès de son environnement utilisateur. L'environnement utilisateur va alors faire une connexion SOAP sécurisée vers la source à l'URL spécifiée en paramètre de la requête GET précédente. La connexion HTTPS permet à la source de présenter son

certificat à l'environnement utilisateur. L'environnement utilisateur informe l'utilisateur du certificat reçu afin que l'utilisateur le vérifie. Si l'utilisateur accepte ce certificat, l'environnement utilisateur récupère le document de métadonnées. Il suffira pour cela qu'il indique dans une requête SOAP le nonce, identifiant à partir duquel la source déterminera le document de métadonnées à fournir en réponse, notamment en identifiant l'utilisateur auquel il servi le nonce. L'environnement utilisateur pourra ensuite rediriger le navigateur de l'utilisateur vers la source en utilisant l'URL de retour fournie en paramètre de la redirection initiale.

(Le fichier de description possède un identifiant. Lorsqu'un certificat est obtenu celui-ci peut contenir l'identifiant de fichier de description. Ce qui permet d'indiquer qu'il est nécessaire de renouveler ce fichier.)

Le schéma suivant résume ce fonctionnement :

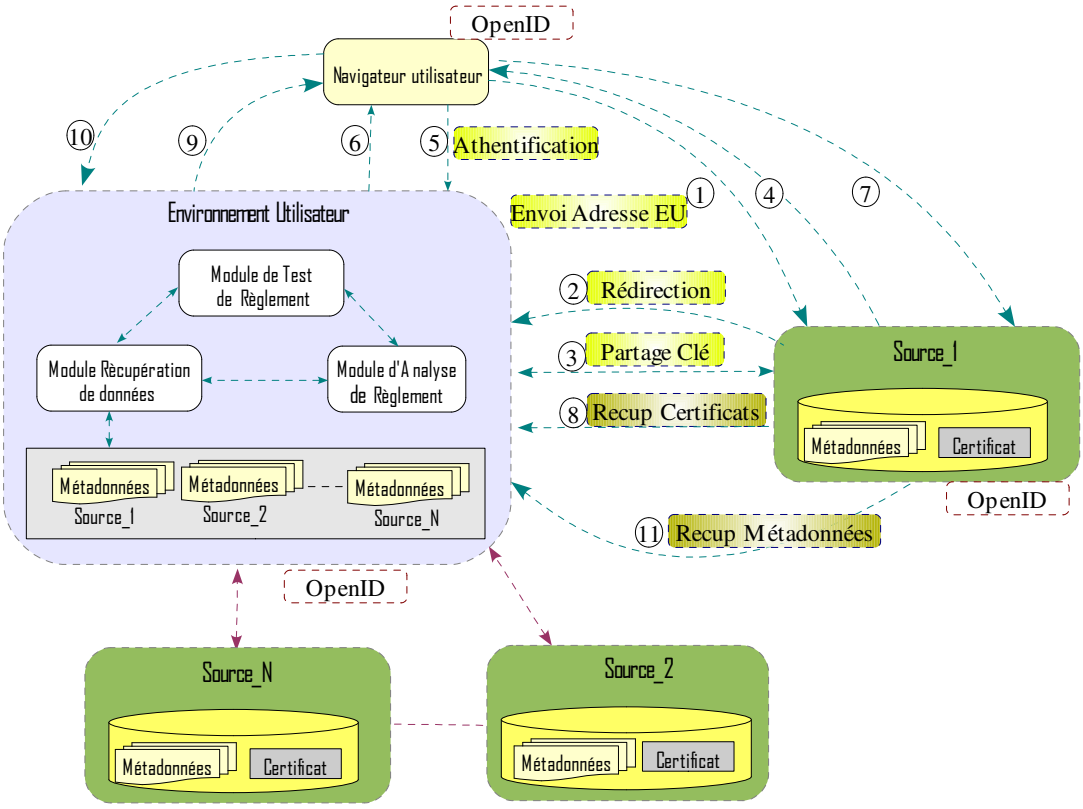

Figure 3.2 : Récupération de la description de source

1. Envoi de l'adresse de l'environnement utilisateur : http://www.mon_eu.org;
2. Redirection vers l'environnement utilisateur pour vérification de preuve d'identités;
3. Partage de clé entre l'environnement utilisateur et la source;
4. La source redirige l'utilisateur vers son environnement en fournissant en GET l'url comme

spécifié ci-dessus;

5. L'utilisateur s'authentifie auprès de son environnement;
6. Son environnement le renvoie vers la source grâce à l'adresse fournie en 4 avec GET;
7. La source vérifie que les données renvoyées sont authentiques;
8. Récupération des certificats par l'environnement utilisateur;
9. Envoi des certificats pour vérification;
10. Réponse d'acceptation;
11. Récupération des métadonnées.

Au cours de cet échange, l'environnement utilisateur peut communiquer un matériel d'authentification à la source. Ça peut être une clé publique RSA par exemple. Ainsi, lorsque l'environnement utilisateur aura besoin de solliciter la source sans interagir avec l'utilisateur pour l'authentification auprès de la source, il utilisera ce matériel d'authentification.

Voici la requête SOAP permettant de récupérer le fichier de métadonnées.

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
  <soap:Envelope
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:disco="urn:metadata:discovery">
        <soap:Body>
              <disco:MetadataRequest>
                    <disco:nonce>_080667A86E27D0B1EF50E29852B2E2EF</disco:nonce>
              </disco:MetadataRequest>
        </soap:Body>
  </soap:Envelope>
```

Réponse contenant le fichier de métadonnées en JSON :

```
<?xml version="1.0" encoding="UTF-8" ?>
  <soap:Envelope
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:disco="urn:MySoapServices">
        <soap:Body>
              <disco:MetadataResponse>
                  <disco:Metadata>
{"SourceName":"attributes_provider.xyz.com","Protocol":"urn:oasis:names:tc:SAML:2.0","Attributes":[{"sn","X500"},{"gn","X500"}]}
                  </disco:Metadata>
              </disco:MetadataResponse>
        </soap:Body>
  </soap:Envelope>
```

3.4.4 Envoi du règlement

Ce protocole est très similaire à celui de récupération du fichier de métadonnées sauf que l'environnement utilisateur récupère le règlement correspondant à la requête d'accès.

Le schéma suivant résume ce fonctionnement :

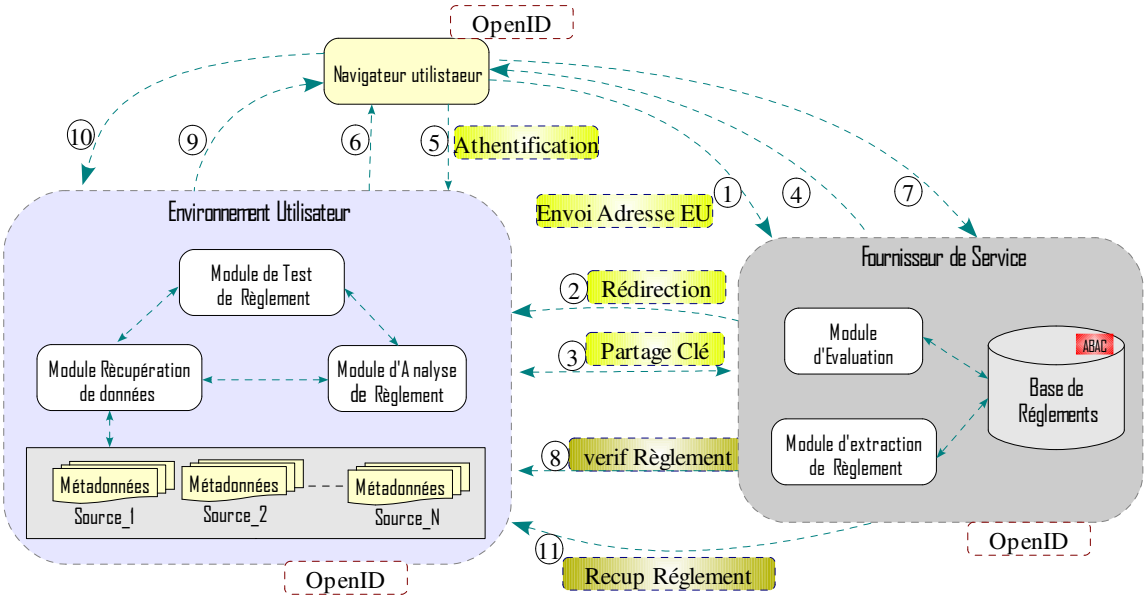

Figure 3.2 : Envoi du règlement

Voici la requête SOAP permettant de récupérer le règlement.

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
  <soap:Envelope
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:disco="urn:metadata:discovery">
        <soap:Body>
              <disco:PolicyRequest>
                    <disco:nonce>_080667A86E27D0B1EF50E29852B2E2EF</disco:nonce>
              </disco:PolicyRequest>
        </soap:Body>
  </soap:Envelope>
```

Réponse contenant le règlement :

```
<?xml version="1.0" encoding="UTF-8" ?>
  <soap:Envelope
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:disco="urn:MySoapServices"
   xmlns:xacml="urn:oasis:names:tc:xacml:2.0:policy:schema:os">
        <soap:Body>
              <disco:PolicyResponse>
                          <xacml:Policy PolicyId="_089852B2E2EF0667A86E27D0B1EF50E2">...</xacml:Policy>
              </disco:PolicyResponse>
        </soap:Body>
  </soap:Envelope>
```

3.4.5 Envoi des certificats

Une fois que l'environnement utilisateur a collecté les certificats (des assertions SAML2) il faut les faire parvenir au fournisseur de service en réponse au règlement. Pour cela, nous allons utiliser un protocole similaire à celui défini précédemment.

L'environnement utilisateur redirige le navigateur de l'utilisateur vers le fournisseur de services en indiquant l'identifiant du règlement (la valeur de l'attribut PolicyId du règlement). Un seul champs supplémentaire contenant une URL est nécessaire. Celle-ci indique l'adresse à laquelle le fournisseur de services récupère un document contenant toutes les assertions obtenues via un échange SOAP.

Le navigateur obtient de l'environnement utilisateur un code de réponse HTTP 302 avec dans l'entête HTTP le champs *Location* suivant :

```
http://fournisseur_de_service.org/requestReturn?
policyId=_089852B2E2EF0667A86E27D0B1EF50E2&url_delivery=http://environnement_utilisateur.org/delivery
```

Voici la requête SOAP permettant au SP de récupérer auprès de l'environnement utilisateur les assertions SAML2.

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
  <soap:Envelope
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:disco="urn:metadata:discovery">
        <soap:Body>
              <disco:CertificateRequest>
                    <disco:policyId>_089852B2E2EF0667A86E27D0B1EF50E2</disco:policyId>
              </disco:CertificateRequest>
        </soap:Body>
  </soap:Envelope>
```

Réponse contenant les assertions :

```
<?xml version="1.0" encoding="UTF-8" ?>
  <soap:Envelope
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:disco="urn:MySoapServices"
   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
        <soap:Body>
              <disco:CertificateResponse>
                        <disco:CertificateBundle>
                    <saml:Assertion>...</saml:Assertion>
                    <saml:Assertion>...</saml:Assertion>
                  </disco:CertificateBundle>
              </disco:CertificateResponse>
        </soap:Body>
  </soap:Envelope>
```

3.5 Conclusion

Dans ce chapitre nous avons traité de l'architecture protocolaire du système. Nous avons défini un système global où nous supposons employer le protocole OpenID, pour résoudre les problèmes d'authentifications multiples, et le protocole SAML pour la récupération des certificats. Ces protocoles étant connus, nous ne les avons pas décrits. Le système global a ainsi été défini en abordant les différentes facettes que sont les espaces de noms et le formatage standard des règlements en XACML. Ce fonctionnement global, exploitant au maximum les standards du Web, est notre première contribution dans ce chapitre. Nous avons ensuite défini plusieurs protocoles complémentaires afin de réaliser cette architecture ce qui constitue notre contribution principale dans ce chapitre et décrite également dans [Ates & Abakar 2011].

Au prochain chapitre nous mettons en œuvre cette architecture.

CHAPITRE 4

IMPLEMENTATION

4 IMPLEMENTATION

4.1 Introduction

L'architecture protocolaire a été définie au chapitre précèdent. Un assemblage de plusieurs protocoles pour la mise en œuvre de notre système a été défini. Ce chapitre traite l'aspect pratique du sujet. Nous avons choisi le langage python et son « framework » Django pour l'implémentation de notre système. Python est un langage portable, dynamique, extensible, et surtout, libre. C'est un langage simple et léger par rapport à d'autres langages, C ou java par exemple [Python Fundation, 2012]. Python est doté d'une librairie standard et extensible qui fait de lui un langage attractif. Les librairies python offrent accès à une grande variété de services y compris pour les protocoles Internet.

Nous utilisons Python pour implémenter un modèle de données permettant la définition de règlements ce qui permettra l'exploitation de nos algorithmes d'analyse en Python en s'appuyant sur ce modèle de donnée déjà introduit au chapitre 2 lors de la création des algorithmes. Enfin, nous avons implémenté un outil de conversion de règlement XACML en objets de ce modèle.

Python est très utilisé dans les développements d'applications Web grâce au « framework » Django. C'est un outil disposant d'une communauté importante et de nombreux modules, notamment des modules facilitant la réalisation d'un espace utilisateur avec des droits d'accès aux données, la réalisation d'un module d'administration et de gestion des sites. Le principe de fonctionnement de Django est proche du modèle MVC (Modèle-Vue-Contrôleur) [Django Fundation, 2012]. Cependant, Django organise la répartition des codes suivant les classes d'applications. On distingue ainsi les entités *Modèle*, les entités *Vue* et et les entités *« Template »* (MVT) en sus du code métier de l'application.

- L'entité modèle permet la modélisation des données et la gestion du stockage.
- Les templates, ou gabarit, sont les modèles de page de l'application. Un langage dédié au « template » est utilisé afin de décrire les variables au sein des templates.
- Les vues constituent les interfaces entre les utilisateurs et l'application. Elles reçoivent les requêtes et réponses HTTP. Elles retournent des réponses HTTP, éventuellement en s'appuyant sur un « template ».

Notre modèle de données est donc crée en Python mais en utilisant les classes de base de Django afin de pouvoir stocker ces objets en base via les outils Django.

Nous présentons ensuite l'implémentation des protocoles présentés au chapitre précédent en utilisant les outils Django. Nous définissons le plan d'URL de chaque rôle de notre architecture. Nous associons les vues aux URLS.

4.2 Modèles de données

Nous avons défini plusieurs algorithmes d'analyse des règlements au chapitre précédent. Le premier algorithme permet la vérification de la disponibilité des informations auprès des sources. Le second permet la vérification des règles. Le troisième permet l'évaluation d'une expression logique. Nous présentons dans cette section le modèle de données utilisé par ces algorithmes ainsi que la création d'un règlement à l'aide de ces classes.

La définition de notre modèle regroupe les six classes principales suivantes :

```
class AssertionData(models.Model)
class AssertionDefinition(models.Model)
class Predicate(models.Model)
class PredicateRequired(models.Model)
class PredicateComparison(models.Model)
class AbacRule(models.Model)
```

Ces classes sont toutes héritées de la classe *Model* de Django. Cela permettra de faire de ces classes des tables de la base de données. Les instanciations de ces classes seront des entrées de ces tables en base.

4.2.1 Expression logique

Nous définissons une entité *AbacRule* avec deux champs. Un champ "expression" qui est une chaîne de caractères représentant l'expression logique. Dans cette expression, les prédicats sont remplacés par un identifiant. A titre d'illustration, considérons la règle logique "age>18 et nationalité=française". Nous avons deux prédicats auxquels sont attribués un identifiant. Par exemple, « 123 » est le prédicat « age>18 » et « 456 » est le prédicat « nationalité=française ». L'expression logique est donc ensuite transformée en « 123 et 456 ».

La classe *AbacRule* possède un second champ qui est destiné à recevoir les prédicats. Les prédicats sont ainsi stockés de manière à posséder l'identifiant utilisé dans l'expression logique. Comme cette classe constitue une table de la base de données, il est nécessaire de stocker les prédicats, qui sont des instances de la classe *Predicate* définie plus tard sous la forme d'une chaîne de caractère. Pour cela nous utilisons les fonctions *dumps()* et *loads()* de la librairie python qui permettent respectivement de transformer une instance en chaîne et une chaîne en instance.

```
class AbacRule(models.Model):
    expression = models.CharField(max_length = 2048, null=True, blank=True)
    predicates = models.CharField(max_length = 4096, null=True, blank=True)

    def get_predicates(self):
        if not self.predicates:
            return []
        return loads(str(self.predicates))

    def add_predicate(self, predicate=None):
        if predicate:
            predicates = self.get_predicates()
            predicates.append(predicate)
            self.predicates = dumps(predicates)
```

4.2.2 Source

Comme défini précédemment, une source de confiance est une entité possédant les trois attributs : « nom », « clé publique » ou « certificat » et « type ».

Les instances de cette classe seront notamment utilisées pour définir la source de confiance attendue pour la délivrance d'une information certifiée. Ainsi, cette source peut être directement définie, son nom est donné, son type est « DIRECT ». Il est possible que la source ne soit pas connue mais soit tout de même de confiance parce qu'il existe une source connue qui est de confiance. Il s'agit donc de spécifier le nom de la source connue et d'indiquer qu'il s'agit d'une source « INDIRECT ». En outre le type SELF indique que l'attribut d'identité n'a pas besoin d'être certifié par une source de confiance.

L'attribut « LOCAL » sert à déclarer une source locale d'attributs d'identité comme un annuaire LDAP ou une base de données par exemple.

```
SOURCE_TYPE = (
    ('DIRECT', _('Direct trusted source')),
    ('ANCHOR', _('Trust anchor')),
    ('LOCAL', _('Local source')),
    ('SELF', _('Untrusted or user self-asserted')))


class Source(models.Model):
    name = models.CharField(max_length = 100, unique=True)
    public_key_or_ssl_certificate = models.TextField(blank=True)
    type_source = models.CharField(
            max_length = 60, choices = SOURCE_TYPE,
            verbose_name = '',
            default = 'DIRECT')
```

4.2.3 Attribut

Un attribut est définit par une définition. La définition définit principalement le nom de l'attribut et son type (chaîne, entier, adresse IP, etc.).

Un attribut peut avoir une ou plusieurs valeurs. La classe *AttributeData* permet de définir des valeurs d'attributs. L'attribut est indiqué par sa définition. Le champ source est ensuite utilisé pour déclarer d'où provient la valeur.

4.2.4 Assertions

Une assertion permet de lier une définition d'attributs ou des valeurs d'attributs à une source. Une assertion de définition (*AssertionDefinition*) sera par exemple utilisée pour définir un prédicat indiquant que l'on souhaite qu'un attribut provienne d'une ou de plusieurs sources, et éventuellement imposer une contrainte sur sa ou ses valeurs. Une instance de la classe *AssertionData* permettra de déclarer des valeurs d'attributs dans les prédicats.

```
class AssertionDefinition(models.Model):
    def __init__(self, definition=None, sources=[]):
        self.definition = definition
        self.sources = []
        for source in sources:
            self.add_source(source)

    def add_source(self, source):
        if not source.id in self.sources:
            self.sources.append(source.id)

    def remove_source(self, source):
        if source.id in self.sources:
            self.sources.pop(source.id)

    def get_sources(self):
        result = []
        for key in self.sources:
            try:
                source_ = AttributeSource.objects.get(id=key)
                result.append(source_)
            except:
                pass
        return result

class AssertionData(models.Model):
    def __init__(self, attribute_data=None):
        self.attribute_data = attribute_data

    def get_attribute_data(self):
        return self.attribute_data

    def set_attribute_data(self, attribute_data):
        self.attribute_data = attribute_data
```

4.2.5 Prédicats

Un prédicat exprime une contrainte sur un ou plusieurs attributs, leurs valeurs, et sur les sources des ces attributs.

- AssertionDefinition est utilisé lorsque le prédicat porte sur un attribut.
- AssertionData est utilisé lorsque le prédicat porte sur la ou les valeurs d'un attribut.

Nous définissons une classe parente pour tous les prédicats :

```
class Predicate(models.Model):
    rule = models.ForeignKey('AbacRule')
```

Le prédicat peut être soit de type « requis » (PredicateRequired) soit de type « comparaison » (PredicatComparaison).

PredicateRequired prend comme champ un objet de type AssertionDefinition :

```
class PredicateRequired:
    def __init__(self, assertion_definition=None, single_value=False,
sources=[]):
        self.assertion_definition = assertion_definition
        self.single_value = single_value
        self.sources = []
        for source in sources:
            self.add_source(source)

    def get_assertion_definition(self):
        return self.assertion_definition

    def set_assertion_definition(self, assertion_definition=None):
        self.assertion_definition = assertion_definition

    def get_definition(self):
        if self.assertion_definition:
            return self.assertion_definition.definition
        return None

    def get_sources(self):
        definition = self.assertion_definition
        if definition:
            return definition.get_sources()
        return None
```

Le prédicat de comparaison, s'applique de la même façon à n'importe quel type de données.

Les deux attributs comparés peuvent être définis à l'aide de deux définitions d'attribut différentes.

Nous définissons donc un unique prédicat pour la comparaison, incluant les tests d'égalité et d'inégalité. La fonction d'évaluation utilisera le paramètre type de ce prédicat pour déterminer le traitement. Ce prédicat a deux champs : « operande1 » et « operande2 ».

```
class PredicateComparison:
    def __init__(self, operand1=None, operand2=None,
            operand1_single_value=False, operand2_single_value=False,
            comparison_type=ACS_XACML_COMPARISON_EQUALITY_STRING,
            multivalues='NO_MULTIVALUES',
            multivalues_explanation=None):
        self.operand1 = operand1
        self.operand2 = operand2
        self.operand1_single_value = operand1_single_value
        self.operand2_single_value = operand2_single_value
        self.comparison_type = comparison_type
        self.multivalues = multivalues
        self.multivalues_explanation = multivalues_explanation

    def get_operand1(self):
        return self.operand1

    def set_operand1(self, operand1=None):
        self.operand1 = operand1

    def get_operand2(self):
        return self.operand2

    def set_operand2(self, operand2=None):
        self.operand2 = operand2
```

4.2.6 Constitution d'un règlement

Une permission ABAC possédant trois champs : un objet/une vue, une action/activité et une règle ABAC. Nous restreignons la notion de règlement à la déclaration d'un ensemble de permissions.

Nous avons repris la notion de généricité introduite dans OrBAC pour les objets et les actions. Une permission peut ainsi porter sur une vue et une activité, qui indiquent respectivement un ensemble d'objets et un ensemble d'actions.

```
class Action(models.Model):
    name = models.CharField(max_length = 40)

class Activity(models.Model):
    name = models.CharField(max_length = 40)
    actions = models.ManyToManyField(Action, verbose_name=_('actions'),
        blank=True)
    activities = models.ManyToManyField('self', symmetrical=False,
        verbose_name=_('activities'), blank=True)

class AcsObject(models.Model):
    name = models.CharField(max_length = 40)
    namespace = models.ForeignKey(Namespace, verbose_name = _('Namespace')

class View(models.Model):
    name = models.CharField(max_length = 40)
    namespace = models.ForeignKey(Namespace, verbose_name = _('Namespace'))
   acs_objects = models.ManyToManyField(AcsObject,
        verbose_name=_('acs objects'), blank=True)
    views = models.ManyToManyField('self', symmetrical=False,
```

```
        verbose_name=_('views'), blank=True)

class AcsAbacPermission(models.Model):
    content_type_what = models.ForeignKey(ContentType, related_name =
"what_abac")
    content_type_how = models.ForeignKey(ContentType, related_name =
"how_abac")
    object_id_what = models.PositiveIntegerField()
    object_id_how = models.PositiveIntegerField()
    what = generic.GenericForeignKey(ct_field='content_type_what',
                                     fk_field='object_id_what')
    how = generic.GenericForeignKey(ct_field='content_type_how',
                                     fk_field='object_id_how')
    rule = models.ForeignKey(AbacRule, verbose_name = _('AbacRule'))
```

A titre d'illustration nous créons une instance de la classe AbacRule pour l'exemple suivant :

```
(age of (IdP1 or IdP2) >= 18 and surname of IdP1 == surname of IdP2
```

Pour l'instanciation de la règle ABAC, nous commencerons en premier lieu par la création des sources de confiances IdP1 et IdP2 et des définitions d'assertions. Puis nous créons tous les prédicats. Enfin, nous définissions la règle logique.

- Création des sources s2 pour IdP1, s2 pour IdP2:
- Instanciation d'une règle

```
rule = AbacRule()
rule.save()
```

- Création d'*AssertionDefinition* pour « `surname of IdP1` », « `surname of IdP2` », « `age of IdP1` », « `age of IdP2` »

```
adef_sn1 = AssertionDefinition(definition='surname')
adef_sn1.add_source(s1)
adef_sn2 = AssertionDefinition(definition='surname')
adef_sn2.add_source(s2)
adef_age1 = AssertionDefinition(definition='age')
adef_age1.add_source(s1)
adef_age2 = AssertionDefinition(definition='age')
adef_age2.add_source(s2)
```

- Création de la valeur « 18 » :

```
val18 = AttributeData(definition='age', values=(str(18),))
val18_d = AssertionData()
val18_d.set_attribute_data(val18)
```

- Les prédicats :

```
p1 = PredicateComparison(operand1=adef_age1, operand2=val18_d,
    comparison_type=ACS_XACML_COMPARISON_INTEGER_GRT_OE,
    operand1_single_value=True, operand2_single_value=True)
rule.add_predicate(p1)
p2 = PredicateComparison(operand1=adef_age2, operand2=val18_d,
    comparison_type=ACS_XACML_COMPARISON_INTEGER_GRT_OE,
    operand1_single_value=True, operand2_single_value=True)
rule.add_predicate(p2)
p3 = PredicateComparison(operand1=adef_sn1, operand2=adef_sn2,
    comparison_type=ACS_XACML_COMPARISON_EQUALITY_STRING_IGN_CASE,
    multivalues='EQUAL_OP1_SUBSET_OP2')
rule.add_predicate(p3)
```

- Expression logique ((p1 ou p2) et p3) :

```
str_rule = "(1|2)&3"
rule.expression=str_rule
rule.save()
```

4.3 Du XACML aux objets python

Nous présentons un algorithme qui permet de convertir les règlements ABAC au format XACML en une règle ABAC présentée au chapitre précédent.

4.3.1 Parcours du document XACML

Nous utilisons la librairie python « ndg_python » pour lire les règlements XACML. Cette librairie implémente diverses fonctions relatives à XACML mais nous ne l'utilisons que pour parcourir les règlements XACML et en extraire le contenu pour créer notre règlement en utilisant notre modèle de données.

Les commandes suivantes permettent de lire un règlement XACML et de disposer du contenu en objets python :

```
from ndg.xacml.core.policy import Policy
from ndg.xacml.parsers.etree.factory import ReaderFactory
PolicyReader = ReaderFactory.getReader(Policy)
policy = PolicyReader.parse('policy.xml')
```

La condition de la première règle est contenue dans l'objet suivant :

```
policy.rules[0].condition
```

4.3.2 Création d'une instance « AbacRule » à partir d'une condition XACML

Le parcours de la condition XACML doit nous permettre de créer les attributs de la règle ABAC que sont l'expression logique et les prédicats de cette expression.

Il s'agit d'un traitement récursif de l'arbre XML dont la racine est la condition. Soit le nœud rencontré est un prédicat et il s'agit d'une feuille de l'arbre, soit il s'agit d'un élément logique (ET, OU ou NON), auquel cas, plusieurs branches partent de ce nœud.

L'analyse d'un "nœud logique" déclenche une récursivité, sinon le nœud est traité comme un prédicat.

Le traitement du "nœud prédicat" crée les objets de notre modèle de données dont l'instance d'un prédicat. Ce prédicat est ajouté à la liste des prédicats de la règle ABAC. La fonction de traitement retourne alors l'identifiant du prédicat.

Le type de "nœud prédicat" est identifié par le *FunctionId* du nœud *Apply* correspondant. S'il n'y a pas de *FunctionId* nous supposons qu'il s'agit d'instancier un objet de la classe *PredicateRequired*. Le nœud Apply contiendra un unique élément `SubjectAttributeDesignator` indiquant le nom de l'attribut et la source.

Le *FunctionId* indique sinon le type de comparaison. Il s'agit donc d'instancier un objet de la classe *PredicateComparison*. La valeur de l'attribut *FunctionId*permet de définir le paramètre du type de comparaison de l'objet.

Enfin, les nœuds fils permettent de déterminer s'il s'agit de comparer un attribut à une valeur, par exemple :

```
<SubjectAttributeDesignator
    AttributeId="age"
    DataType="http://www.w3.org/2001/XMLSchema#integer"
    Issuer="IdP3"/>
<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">
    <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer">18</AttributeValue>
</Apply>
```

ou deux attributs, par exemple :

```
<SubjectAttributeDesignator
    AttributeId="urn:ndg:security:authz:1.0:surname"
    DataType="http://www.w3.org/2001/XMLSchema#string"
    Issuer="IdP1"/>
<SubjectAttributeDesignator
    AttributeId="urn:ndg:security:authz:1.0:surname"
    DataType="http://www.w3.org/2001/XMLSchema#string"
```

```
    Issuer="IdP2"/>
```

Voici donc un aperçu de cette méthode :

```
def treat_predicate(predicate_node, abac_rule):
    predicate = …
    abac_rule.add_predicate(predicate)
    return str(predicate.id)
```

L'analyse d'un « nœud logique » se fait par la fonction « treat_rule() ». Cette fonction récupère les identifiants des prédicats pour ses nœuds fils de type prédicat et s'appelle pour ses nœuds fils de type « nœud logique ». Elle retourne une chaîne de caractère correspondant à l'expression logique :

```
LOGIC_OPERATORS = {"urn:oasis:names:tc:xacml:1.0:function:and": '&',
    "urn:oasis:names:tc:xacml:1.0:function:or": '|',
    "urn:oasis:names:tc:xacml:1.0:function:not": '-'}

def treat_rule(rule, abac_rule):
    if not rule.functionId in LOGIC_OPERATORS.keys():
        return treat_predicate(rule, abac_rule)
    else:
        children = [treat_rule(r, abac_rule) for r in rule.expressions]
        exp = '('
        for child in children:
            exp += child + LOGIC_OPERATORS[rule.functionId]
        return exp[:-1] + ')'

PolicyReader = ReaderFactory.getReader(Policy)
policy = PolicyReader.parse('policy.xml')
apply_racine = policy.rules[0].condition.expression

abac_rule = AbacRule()
abac_rule.expression = treat_rule(apply_racine, abac_rule)
```

4.4 Implémentation des protocoles en Django

Nous présentons ensuite l'implémentation des protocoles présentés au chapitre précédent en utilisant les outils Django. Nous définissons le plan d'URL de chaque rôle de notre architecture. Nous associons les vues aux URLS et nous expliquons brièvement ce que contiennent les vues.

4.4.1 Redirections et requêtes SOAP en Django

Une vue reçoit comme paramètre d'entrée une requête HTTP et retourne une réponse HTTP. L'association des vues aux URLs se gère en Django avec des *urlpatterns*.

Des objets Django permettent de gérer simplement les requêtes et les réponses. Les requêtes sont gérées par la classe `HttpRequest`. La réponse se gère à l'aide de la classe `HttpResponse`. Il existe également une classe héritée de `HttpResponse` qui permet de gérer les redirections soit `HttpResponseRedirect`.

Pour les appels SOAP nous utilisons la librairie « urllib » :

```
import urllib

def soap_call(url, msg):
    conn = httplib.HTTPConnection(host)
    conn.request('POST', query, msg, {'Content-Type': 'text/xml'})
    response = conn.getresponse()
    data = response.read()
    conn.close()
    return data
```

Nous n'utilisons pas de complexité particulière liée à SOAP et nous créons donc manuellement les messages SOAP :

```
msg = '''<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body>%s</soap:Body>
</soap:Envelope>''' % content
```

La réception d'un message SOAP se fait tout aussi simplement :

```
def url_to_receive_SOAP_request(request):
    soap_message = None
    if request.method == 'POST' and \
            'text/xml' in request.META['CONTENT_TYPE']:
        soap_message = request.raw_post_data
```

4.4.2 Plan d'adressage

Il s'agit ici de déterminer, pour les protocoles définis au chapitre précédent, les points d'entrée applicatifs pour chaque rôle des protocoles, puis de leur associer une URL. Cette URL est ensuite associée à la vue définie pour le traitement du message protocolaire.

Voici les protocoles que nous souhaitons traiter :

- Déclaration de sources d'attributs auprès de l'environnent utilisateur (Protocole P1) : implique un environnement utilisateur (EU) et une source (S)
- Envoi du règlement (Protocole P2) : implique un environnement utilisateur (EU) et un fournisseur de services (SP)
- Envoi des certificats (Protocole P3) : implique un environnement utilisateur (EU) et un fournisseur de services (SP)

Phase protocolaire de P1	Direction	URL sur la destination	Vue sur destination
Initiation de P1	S vers EU	/declaration_source/initiation	views.declaration_source.initiation
Connexion SOAP de P1	EU vers S	/declaration_source/storage	views.declaration_source.storage
Retour sur la source de P1	EU vers S	/declaration_source/retour	views.declaration_source.retour

Phase protocolaire de P2	Direction	URL sur la destination	Vue sur destination
Initiation de P2	SP vers EU	/envoi_reglement/initiation	views.envoi_reglement.initiation
Connexion SOAP de P2	EU vers SP	/envoi_reglement/storage	views.envoi_reglement.storage
Retour sur la source de P2	EU vers SP	/envoi_reglement/retour	views.envoi_reglement.retour

Le protocole P3 à la particularité d'être déclenché lors du retour de P2. Nous utilisons donc la même vue et la même URL du SP. En outre, il n'y a pas de retour vers EU puisque l'utilisateur va consommer le service sur le SP.

Phase protocolaire de P3	Direction	URL sur la destination	Vue sur destination
Initiation de P3	EU vers SP	/envoi_reglement/retour	views.envoi_reglement.retour
Connexion SOAP de P3	SP vers EU	/envoi_certificats/storage	views.envoi_certificats .storage

En Django, la déclaration de URLs se fait à l'aide d'objets *patterns*. Voici le fichier URL de la source pour le protocole P1.

```
from django.conf.urls.defaults import *

urlpatterns = patterns('',
    (r'^declaration_source/storage/', 'views. declaration_source.storage'),
    (r'^declaration_source/retour', 'views. declaration_source.retour'),
)
```

Voici le fichier URL de l'environnement utilisateur pour les protocoles P1, P2 et P3.

```
from django.conf.urls.defaults import *

urlpatterns = patterns('',
```

```
    (r'^declaration_source/initiation/', 'views.
declaration_source.initiation'),
    (r'^envoi_reglement/initiation/', 'views.
declaration_source.initiation'),
    (r'^envoi_certificats/storage/', 'views. envoi_certificats .storage'),
)
```

Voici le fichier URL du fournisseur de services pour les protocoles P2 et P3.

```
from django.conf.urls.defaults import *

urlpatterns = patterns('',
    (r'^envoi_reglement/storage/', 'views.envoi_reglement.storage'),
    (r'^envoi_reglement/retour', 'views.envoi_reglement.retour'),
)
```

4.4.3 Traitement des requêtes

Django permet un traitement simplifié des requêtes HTTP ce qui permet d'extraire les paramètres de l'URL simplement.

En section 4.4.1 nous avons déjà montré comment extraire la requête SOAP de la requête HTTP.

Pour récupérer un paramètre, par exemple le champ nonce, voici l'instruction à utiliser :

```
nonce = request.REQUEST.get('nonce')
```

4.5 Interfaces graphiques

4.5.1 Coté Environnement utilisateur

Au niveau de l'environnement utilisateur, nous disposons des interfaces d'interaction avec le user. Ces interfacent disposent des moyens permettant :

- l'authentification de l'utilisateur ;
- l'enregistrement des métadonnées des sources ;
- de montrer à l'utilisateur le résultat de l'analyse du règlement ;
- de lui demander ses choix de sources ;
- de lui demander s'il accepte ses informations personnelles ;
- de l'informer des informations qui seront diffusées.

Pour chacune de ces pages il s'agit de faire un plan d'adressage comme cela a été fait dans une section précédente. Il faut ensuite définir un fichier de style au format CSS. Les pages HTML sont générées à partir de « template » Django.

A titre d'exemple, voici le « template » d'authentification de l'utilisateur par login et mot de passe :

```
{% load i18n %}
<div id="content">
<div>
<form method="post" action="">
{% csrf_token %}
{{ form.as_p }}
<input type="submit" name="{{ submit_name }}" value="{% trans "Log in" %}"/>
{% if cancel %}
<input type="submit" name="cancel" value="{% trans 'Cancel' %}"/>
{% endif %}
</form>
</div>
</div>
```

Les variables sont notées :

```
{% nom_variable %}
```

Celles-ci sont données au « template » via un objet contextuel lorsque la vue retourne la réponse HTTP. Pour faire cela, Django fournit la fonction *render_to_response*.

4.6 Conclusion

Pour l'implémentation nous avons porté notre choix sur l'utilisation du langage python et son « framework » Django. Ce choix a permis d'une part une conception de notre modèle de données et d'autre part une définition des points d'entrée applicatifs par l'utilisation des vues et du plan d'adressage.

Du coté fournisseur de service, les règlements de la politique de contrôle d'accès sont exprimés avec notre modèle puis converti en langage XACML. A l'inverse, côté utilisateur il faut un algorithme de retranscription de ces règlements en objets python. Les objets obtenus servent aux algorithmes implémentés et introduits au chapitre 2.

Pour l'interopérabilité, Django dispose des outils permettant de gérer le protocole HTTP ce qui a optimisé la phase d'implémentation des protocoles.

Il reste dans une deuxième phase à intégrer ces développements dans les applications Authentic2 et Veridic pour le déploiement en exploitant les protocoles d'obtention et de diffusion standards de certificats. Les interface graphiques d'interaction avec l'utilisateur sont enfin à concevoir et à intégrer à Authentic2.

La conception d'un système de contrôle d'accès a été validé par les implémentations

présentées. Cette première phase d'implémentation est une contribution par rapport aux implémentations existantes portant sur le contrôle d'accès en environnement ouvert.

CONCLUSION

Conclusion

De nos jours, beaucoup d'organisations œuvrent pour la mise en place de systèmes de gestion des identités numériques. Les architectures de fédération d'identités telles qu'initialement définies par Liberty Alliance, Shibboleth ou Passeport de Microsoft sont des exemples types à citer. Nous avons montré dans nos travaux que toutes ces architectures sont exploitées dans des contextes où l'usager qui demande un service auprès d'un fournisseur de service doit être préalablement enregistré auprès d'un fournisseur d'identité.

Actuellement, beaucoup d'entreprises ou d'organisations prestataires de services offrent leurs services en ligne en se limitant à l'exploitation de la délégation d'authentification de la fédération d'identité. Cependant, les transactions en environnement ouvert requièrent des informations plus nombreuses et diverses. Nous avons donc conçu un système répondant à la problèmatique de contrôle d'accès en environnement ouvert. Le système que nous avons conçu est beaucoup plus général que ceux existant dans la mesure où l'utilisateur lui même collecte ses certificats et les diffuse pour obtenir des services. Notre architecture d'authentification est centré sur l'utilisateur qui grâce à son environnement riche en ligne échange avec ses interlocuteurs pour collecter ses certificats et les diffuser.

Pour atteindre notre objectif, nous avons reparti les taches en quatre chapitres.

Au premier chapitre nous avons débiter par l'étude de cas d'usage mettant en évidence les problématiques de la gestion d'identité numérique en environnement ouvert. Ensuite, nous avons fait une étude sur les organisations virtuelles, la notion de partenariat entre organisations et la confiance. La collaboration et l'échange d'informations entre organisations nécessitent l'établissement de liens de confiance entre les différentes entités intervenants. Nous avons ensuite déterminé que le modèle de contrôle d'accès adapté à notre besoin est le contrôle d'accès d'accès basé sur les attributs. Nous l'avons exploité de manière à traiter la problématique de liens de confiance.

Dans le second chapitre, nous avons étudié la description fonctionnelle de notre système construit autour d'un environnement utilisateur riche en ligne. L'utilisateur exprime sa demande au fournisseur de service qui analyse sa requête et lui envoie un règlement à satisfaire. L'utilisateur analyse le règlement et récupère les certificats requis. Il les diffuse ensuite au fournisseur de service pour obtenir l'accès. L'environnement utilisateur dispose de trois modules principaux : un module d'analyse de règlement, un module de récupération des données et un module de validation de

règlement. Le module d'analyse de règlement a pour rôle d'analyser le règlement envoyé par le fournisseur de service. Ce module utilise l'algorithme de recherche de disponibilité de données de l'utilisateur auprès des sources. Le module de récupération des données récupère les descripteurs des sources (les métadonnées) et les certificats à partir des différentes sources de données. Et le module de validation permet de vérifier la satisfaction du règlement. Ce module évalue l'expression logique de la règle en fonction des certificats récupérés. A l'issue de ce test , l'utilisateur décide de diffuser ou non ses certificats au fournisseur de services. Ce dernier reçoit les certificats et utilise ce même algorithme pour déterminer si le règlement est satisfait.

Le chapitre trois a traité l'aspect architectural des protocoles de communication. Nous avons mis en place une architecture protocolaire pour résoudre les problèmes d'échange d'informations.

Enfin, dans le dernier chapitre nous avons traité l'implémentation et la mise en œuvre de notre système. Nous nous sommes basés sur le langage python et son « framework » Django pour implémenter la première partie du système. Il s'agit d'implémenter et tester les différents modules de l'environnement utilisateur à l'aide des différents algorithmes. Les requêtes et les redirections SOAP entre les différentes entités sont implémentées à l'aide de Django. Nous avons établi un plan d'adressage qui a permis de déterminer pour les protocoles des points d'entrées applicatifs. Ainsi nous avons traité les protocoles pour :

- La déclaration de sources d'attributs auprès de l'environnent utilisateur.
- L'envoi du règlement.
- L'envoi des certificats.

Bien que l'intégralité de l'architecture n'est pas été implémentées, la conception de notre système de contrôle d'accès a été validé par les implémentations présentées. Il s'agit d'une première phase qui est une contribution par rapport à l'existant dans le domaine du contrôle d'accès en environnement ouvert.

ANNEXES

Annexes

Les acronymes

Définition	***Page***
AA : Autorité d'Attributs	*25*
ABAC : Attribute based Access control	*17*
CARL : Card-based Access control Requirements Language	*32*
CNIE : Carte nationale d'Identité Numérique	*31*
CoT : Cercle de confiance	13
CSS : Cascading Style Sheets	79
DAC : Discretionary Access Control	15
EU : Environnement Utilisateur	28
HTTP : HyperText Transfer Protocol	53
HTML : HyperText Markup Language	79
IBAC : Identity Based Access Control	15
IMI : Identity Metasystem Interoperability	53
IdP : Identity Provider	23
IDWSF : Identity Web Service Framework	52
IP : Internet Protocol	70
ISIP : Identity Selector Interoperability Profile	53
JSON : JavaScript Object Notation	56
LA : Liberty Alliance	12
LDAP : Lightweight Directory Access Protocol	23
MAC Mandatory Access Control	15
MVC : Modèle-Vue-Controleur	67

Tables des figures

Références bibliographiques

[Abou El Kalam & al, 2003] A. Abou El Kalam, R. El Baida, P. Balbiani, S. Benferhat, F. Cuppens, Y. Deswarte, A. Miège, C. Saurel et G. Trouessin Organization Based Access Control. IEEE 4th International Workshop on Policies for Distributed Systems and Networks (Policy 2003), Lake Come, Italy, June 4-6, 2003.

[Abou El Kalam & Deswarte, 2006] Anas Abou El Kalam, Yves Deswarte. Multi-OrBAC: a New Access Control Model for Distributed, Heterogeneous and Collaborative Systems, 8th IEEE International Symposium on Systems and Information Security (SSI 2006), Sao Paulo, Brésil, 8-10 novembre 2006.

[Andreas & Chris, 2003] Andreas Pashalidis and Chris J. Mitchell. A Taxonomy of Single Sign-On Systems. In Lecture Notes in Computer Science, volume 2727, pages 249 264, Junary 2003.

[Ates, 2008] Ates Mikaël, Gravier Christophe, Fayolle Jacques, Sauviac Bruno, 2008. Marrying Heterogeneous Circles of Trust: No Silver Bullet Yet. Pages 240243 de : IECCS'07: Proceedings of the International Electronic Conference on Computer Science 2007, vol. 1060. AIP Conf. Proc.

[Ates, 2009] Ates Mikaël, Jacques Fayolle, Christophe Gravier, Jeremy Lardon, "The user-centric vision matches credentials exchanges", Proceedings of the forth International Conference on Availability, Reliability and Security 2009 (ARES09), Fukuoka, Japon, March 16-19, 2009, pp 870—876, isbn 978-0-7695-3564-7, doi 10.1109/ARES.2009.62, IEEE Computer Society.

[Ates & Abakar, 2011] Ates Mikaël, Serge Ravet, Mahamat Ahmat Abakar, Jacques Fayolle, "An Identity-Centric Internet: Identity in the Cloud, Identity as a Service and other delights", Proceedings of the Sixth International Conference on Availability, Reliability and Security 2011 (ARES11), Vienna, Austria, August 22-26, 2011.

[Baina & al, 2007] Amin Baina, Yves DESWARTE, Anas Abou El kalam. Approche pour le contrôle d'accès collaboratif dans les infrastructures critiques,

EDSys 2007 – 8e congrès des doctorants, http://edsys2007.enstimac.fr

[Baina, 2009] Amin Baina. Thèse : Contrôle d'Accès pour les Grandes Infrastructures Critiques : Application au réseau d'énergie électrique, Université de Toulouse, septembre 2009.

[Bell & LaPadulla, 1976] D. E. Bell et L. J. LaPadula. Secure computer systems: Unified exposition and multics interpretation. Technical Report ESD-TR-73-306, The MITRE Corporation, Mars 1976.

[Blaze & al, 1996] Matt Blaz, Joan Feigenbaum, Jack Lacy. Decentralized trust management, in Proceedings of the 1996 IEEE Symposium on Security and Privacy pages

164-173

[Bertino & al, 1996] Elisa Bertino, Sushil Jajodia, Pierangela Samarati. Supporting Multiple Access Control Policies in Database Systems IEEE symposium on security and privacy, Oakland, CA, mai 1996, pp 94-107 location : http://www.cacs.louisiana.edu/~jyoon/grad/adb/References/secpap/multi-ac-conflict01tods.pdf

[Bray, 2006] Bray Tim, Hollander Dave, Layman Andrew & Tobin Richard. 2006 (August). Namespaces in XML 1.0 (Second Edition). Tech. rept. W3C World Wide Web Consortium and IETF Internet Society.

[Camarinha-Matos et Lima, 1998] L. Camarinha-Matos, C. Lima. A Framework for Cooperation in Virtual Enterprises, Design of Information Infrastructures Systems for Manufacturing 1998, Fort Worth, USA, May 1998.

[Camenisch & al, 2010] Jan Camenisch, Sebastian Moedersheim, Gregory Neven, Franz-Stefan Preiss, and Dieter Sommer. A Language Enabling Privacy-Preserving Acces Control, *ACM SACMAT 2010*, *pp. 119--128*, ACM

[Capitani & Samarati, 1996] Sabrina de Capitani di Vimercati, Pierangela Samarati. Access Control in Federated Systems, ACM SIGSAC New Security Paradigms Workshop, Lake Arrowhead, CA, 31 août 1996, pp. 87-9

[Clercq, 2002] Clercq Jan De. 2002. Single Sign-On Architectures. Pages 4058 de : InfraSec '02: Proceedings of the International Conference on Infrastructure Security . London, UK: Springer-Verlag.

[Cofta & al, 2007] Cofta Piotr. 2007. Trust, Complexity and Control - Condence in a convergent world.Wiley.

[Coma, 2009] Céline Coma. Thèse : Interopérabilité et cohérence de politiques de sécurité pour les systèmes auto-organisants, Ecole superieure des télécommunications de Bretagne, 22 avril 2009

[Corine Merrini, 2001] Corine Merrini. Le partenariat : Histoire et essai de définition, acte de la journée nationale de l'OZP, 5 mai 2001

[Crokford, 2006] D. Crokford. The application/json Media Type for JavaScript Object Notation, july 2006, http://tools.ietf.org/pdf/rfc4627.pdf

[Cuppens et al, 2006] Frédéric Cuppens, Nora Cuppens-Boulahia, Céline Coma. O2O: Virtual Private Organizations to Manage Security Policy Interoperability, 2nd International Conference on Information Systems Security (ICISS 2006), Calcutta (Inde), 19-21 décembre 2006, Springer, LNCS n°4332, pp. 101-115.

[Django Fundation, 2012] Django Fundation. The web framework for perfectionists with deadlines 2012, https://www.djangoproject.com/foundation/

[Eric & al, 2005] Eric Yuan, Jin Tong. Attribute Based Access Control, A New Access Control Approach for Service Oriented Architectures (SOA), New Challenges for Access Control Workshop, Ottawa, ON, Canada, 2005

[Grob, 2003] Grob Thomas. Security Analysis of the SAML Single Sign-on Browser/Artifact Prole. Page 298 de : ACSAC '03: Proceedings of the 19th Annual Computer Security Applications Conference. Washington, DC, USA: IEEE Computer Society, 2003

[Group Oxford, 2007] Group Oxford Computer. 2007 (February). Achieving Interoperability between Active Directory Federation Services and Shibboleth. Tech. rept.

[Hristo, 2004] Hristo Koshutanski and Fabio Massacci. An Interactive Trust Management and Negotiation Scheme. In Formal Aspects in Security and Trust (FAST'04), Toulouse, France, August 22-27 2004.

[Huu & al, 2005] Huu Tran, Michael Hitchens, Vijay Varadharajan, Paul Watters. A Trust based Access Control Framework for P2P File-Sharing Systems. In 38th Annual Hawaii International Conference on System Sciences (HICSS'05), Hawaii, January 03-06 2005.

[Jajodia & al, 2001] Sushil Jajodia, Pierangela Samarati, Maria Luisa Sapino, V. S. Subrahmanian. Flexible Support for Multiple Access Control Policies, ACM Trans. Database Systems, Vol. 26, no. 2, juin 2001, pp. 214-260.

[Kamel, 2009] Kamel Michel. these Patrons organisationnels et techniques pour la sécurisation des Organisations virtuelles , Université de Toulouse, 29 setembre 2009

[Lampson, 1971] Butler Lampson. Protection. 5th Princeton Symposium on Information Sciences and Systems, pages 437-443, March 1971.

[Lawrence, 2006] Lawrence K. 2006 (February). Web Services Security: SOAP Message Security 1.1 (WS-Security 2004). Tech. rept. Organization for the Advancement of Structured Information Standards.

[Lawrence, 2009] Lawrence Kelvin & Kaler Chris. 2009c (February). Web Services Trust Language (WS-Trust) v1.4. Tech. rept. Organization for the Advancement of Structured Information Standards - Web Services Secure Exchange Technical Committee.

[Lilian Beuzer, 2004] Lilian Beuzer. Complexité : Techniques de calcul et de reduction » Groupe ESIEE Paris 2004, http://www.esiee.fr/~buzerl/ENSEIGNEMENT/IN311/poly_in311.pdf

[Lin, 2006] [Lin, 2006] Linn John. 2006. Trust Models Guidelines. Tech. rept. Organization for the Advancement of Structured Information Standards

[Lorch & al, 2003] Markus Lorch, Seth Proctor, Rebekah Lepro, Dennis Kafura, Sumit Shah. First Experiences Using XACML for Access Control in Distributed

Systems, *ACM Workshop on XML Security*, October 31, 2003, Fairfax, VA, USA

[Maler, 2006] Maler Eve. 2006 (October). Security Assertion Markup Language v2.0 – Technical Overview. Tech. rept. Organization for the Advancement of Structured Information Standards.

[McCabe, 1976] Thomas McCabe. A complexity measure IEEE Transactions on software engineering, vol. SE-2, N04, december 1976

[Nasser, 2006] Bassem Nasser. Organisation Virtuelle : Gestion de politiques de contrôle d'accès inter domaine, Thèse de doctorat, Université Paul Sabatier, Toulouse, France, novembre 2006, 234 pp.

[Nissenbaum, 1998] Nissenbaum Helen. 1998. Protecting Privacy in an Information Age: The problemù of Privacy in Public. Washington Law Review, 17, 559596.

[OASIS, 2005] OASIS. eXtensible Access Control Markup Language (XACML) version 2 OASIS Standard, 1 feb 2005

[OASIS, 2004] OASIS. Trust models guidelines, submitted on the SSTC by Liberty Alliance on 2 febrary 2004, Location : http://www.oasis-open.org/committees/documents.php?wg_abbrev=security

[OASIS, 2010] OASIS. eXtensible Access Control Marup Language (XACML) Version 3.0 Committee Specification-01 10 août 2010, http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-cs-01-en.pdf

[OpenID Community, 2008] OpenID-Community. 2008. OpenID Authentication 2.0 and OpenID Attribute Exchange 1.0 - http://openid.net/developers/specs/.

[Pfitzmann, 2003] Pfitzmann Birgit. 2003. Privacy in Enterprise Identity Federation: Policies for Liberty Single Signon. Pages 189204 de : Lecture Notes in Computer Science: Privacy Enhancing Technologies, vol. 2760. Springer Berlin / Heidelberg.

[Pfitzmann & Waider, 2004] Pfitzmann Birgit & Waidner Michael. 2004. Federated Identity-Management Protocols - Where User Authentication Protocols May Go. Lecture Notes in Computer Science

[python fundation, 2012] PSF Python Software Fundation. 2012, http://www.python.org/psf/

[Rivest & al, 1978] Rivest R. L., Shamir A., Adleman L. 1978. A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM, 21(2), 120126

[Saadi, 2009] Saadi Rachid. 2009. Le contrôle d'accès et la gestion de la conance dans les systèmes pervasifs. INSA Lyon.

[Sandhu & al, R. Sandhu, E. J. Coyne, H. L. Feinstein, C. E. Youman. Role-based access

1996] control models, IEEE Computer, 29(2):38-47, 1996.

[Shehab & al, 2005] Mohamed Shehab, Elisa Bertino, Arif Ghafoor, “Secure collaboration in mediator-free environments”, 12th ACM Conference on Computer and Communications Security (CCS’05), Alexandria, VA, USA, 7-11 novembre 2005, pp. 58-67. http://cobweb.ecn.purdue.edu/~iisrl/publications/pub/ccs-shehab.pdf

[Sturm & al, 2008] Christoph Sturm, Klaus R. Dittrich, Patrick Ziegler. An Access Control Mechanism for P2P Collaborations, DAMAP’08, March 25th, 2008, Nantes, France.

[Suriadi & al, 2007] Suriadi Suriadi, Foo Ernest & Josang Audun. 2007. A User-centric Federated Single Sign-on System. Pages 99106 de : NPC '07: Proceedings of the 2007 IFIP International Conference on Network and Parallel Computing Workshops. Washington, DC, USA: IEEE Computer Society.

[Tourzan & Koga, 2007] Tourzan Jonathan & Koga Yuzo. 2007. Liberty ID-WSF Web Services Framework Overview. Tech. rept. Liberty Alliance Project.

[Upton et Mcafee, 1996] D. M. Upton. et A. Mcafee. The Real Virtual Factory, Harvard Business Review, July-August, 1996.

[Yahalom & al, 1993] Yahalom, R.; Klein, B.; Beth, T. “Trust relationships in secure systems-a distributed authentication perspective“,Research in Security and Privacy, 1993. Proceedings., 1993 IEEE Computer Society Symposium on 24-26 May 1993 Page(s):150 - 164

[Wiederhold, 1992] Gio Wiederhold. Mediators in the Architecture of Future Information Systems, IEEE Computer, vol. 5, no. 3, mars 1992, pp. 38-49.

Wu & Periorellis, 2005] J. Wu and P. Periorellis. Evaluation of Authorization-Authentication Tools: Permis, oasis, xacml et shibboleth, 2005